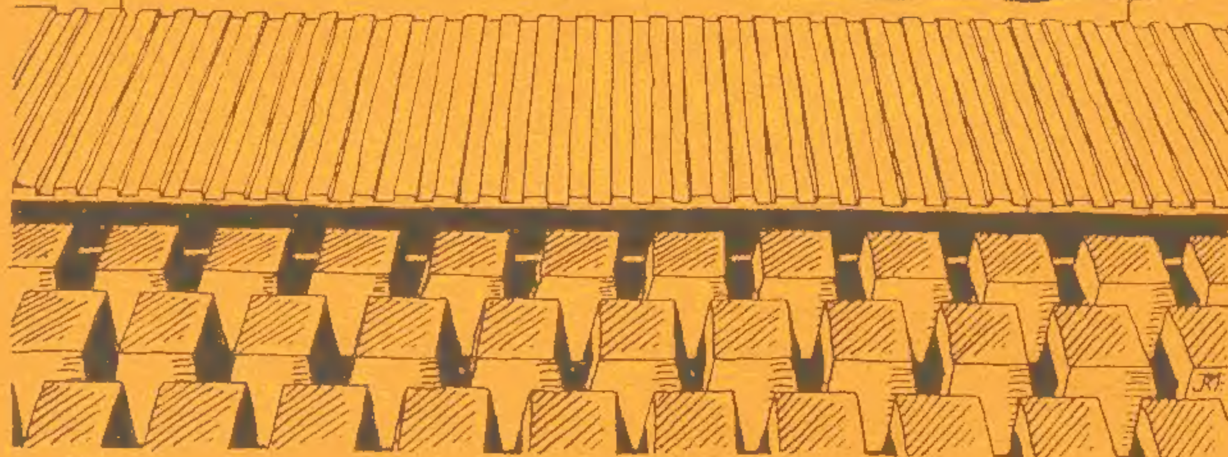
ACORN NIEUWS

JAARGANG 4
NUMMER 2



Bestuur:

Voorzitter:
N. Stad
Plataanweg 47.
1544 PB Zaandijk.
Telf. 075-280808

Secretaris:
N. Schone
Sluispad 55-B
1521 EZ Wormerveer
Telf. 075-286624

Penningmeester:
Th. van Kempen.
Het Puyven 71.
5672 RB Nuenen.
Telf. 040 - 836210

Clubwinkel:
N. Stad
Plataanweg 47
1544 PB Zaandijk
Telf. 075 - 280808

Hard-ware cie.:
P. Ehrlich
Roostenlaan 266
5644 BS Eindhoven.
Telf. 040 - 114183

Redaktie Acorn Nieuws.:
H.Reinders.
Leeuwarderstraat 8.
9718 MX Groningen.
Telf. 050 - 125458

Contributie 1985: fl. 60.00
Leden buitenland fl. 75.00

Giro 5244293 BANK 52.84.69.810
Beide t.n.v. Penningmeester Acorn computerclub.
Nuenen.

Redactie ACORN NIEUWS:
Rudi van Drunen - hardware
Hans Marks - software
Henk Reinders - layout

Redactie adres:
Postbus 1050
9700 BA Groningen
Telf. 050-125458

Ledenadministratie:
S. van Leeuwen
Kompasstraat 32.
1973 PX IJmuiden
Telf 02250-224350

Datasheets:
G. Akkermans.
Wikke 1
1273 BR Huizen.
Telf 02152 - 60294.

Drukwerkarchief:
F. Monsato
Biesbosch 153
8032 VD Zwolle.

Uiterste datum inlevering copy:
NR 3. 8 - 3 - 1985.
nr 4. 19 - 4 - 1985.
NR 5. 5 - 7 - 1985.
NR 6. 16 - 8 - 1985.
NR 7. 27 - 9 - 1985.
NR 8. 8 - 11 - 1985.

DE CLUB-WINKEL:

Geheugenkaart; 16 kByte extra in de ATOM	fl. 40,00
Schakelkaart; meerdere EPROM's op Axxx (incl.74LS133)	fl. 50,00
Programmerkaart; zelf programmeren van EPROMs	fl. 21,00
Big Benny print;incl. Clock IC MSM 5832RS	fl. 42.50
Los IC MSM 5832RS	fl. 32.50
Minischakelkaart	fl. 16.00
Herdruk ACORN NIEUWS 1982; 97 pag. wetenswaardigheden	fl. 6.00
Jaargang 1983; totaal ruim 450 pag.	fl. 30.00
Jaargang 1984;	fl. 35.00
ATOM-WARE deel 1; machinetaal op de ATOM 90 pag.	fl. 6.00
ATOM-WARE deel 2; het diskdrive operatingsystem 60 pag.	fl. 5.00

Bestellen:
Bij Uw regionale penning-meester.
Rechtstreeks bij de federatieve penningmeester, dan fl. 4.00 extra voor porto-kosten.

ACORN NIEUWS is een uigave van de fed. acorn computerclubs n/b
verschijnt 6 - 8 keer per jaar.

Aangezien KOOS OTTES een nieuwe baan heeft gekregen, waarbij hij enige tijd cursus moet volgen, heeft hij zijn bestuursfunctie moeten neerleggen. NICO SCHONE is zodoende als secretaris tot het bestuur toegetreden, terwijl NICO STAD als voorzitter zal fungeren.
De leden administratie zal in het vervolg gedaan worden door SIMON VAN LEEUWEN. De adressen vindt U op pagina 2.

De schakelkaart is momenteel uitverkocht en is alweer in bestelling. Wij verwachten dat deze eind april wederom leverbaar is.
Ook is het tweede deel van ATOMWARE verschenen. Voor de prijs van fl. 5.00 wordt U in 68 pagina's geinformeerd over de DOS-ROM.

Schijf nr 1. 85/2

DISK-V1	2900	AFAF	04F35	002	A.N. nr.2 blz 14.
DBOX-V1	A000	DFFF	01000	052	A.N. nr.2 blz 14.
BURTIME	2900	AFAF	01B79	062	A.N. nr.2 blz 24.
SIMPLEX	2900	AFAF	008D6	07B	A.N. nr.2 blz 28.
IOIS	2900	AFAF	013FD	084	A.N. nr.2 blz 32.
ED64	A000	ACFD	01000	098	A.N. nr.2 blz 34.
SALFAA	A000	A09E	01000	0A8	A.N. nr.2 blz 54.
DECADOS	2900	AFAF	011BA	0B8	A.N. nr.2 blz 36.
PAINTED	2900	AFAF	02CF5	0CA	A.N. nr.2 blz 46.
SPREDIT	2900	AFAF	0311E	0F7	A.N. nr.2 blz 46.
MDCR24R	A000	A000	01000	129	A.N. nr.2 blz 49.
SWITCH	2900	AFAF	01397	139	A.N. nr.2 blz 54.
GAGSDEM	2900	AFAF	009F7	14D	A.N. nr.2 blz 68.
LICHTP	2900	AFAF	00537	157	A.N. nr.2 blz 82.
BOOT-TR	2900	AFAF	0073E	15D	A.N. nr.2 blz 70.
INPUT1	2900	AFAF	00515	165	A.N. nr.2 blz 72.
INPUT2	2900	AFAF	003F1	16B	A.N. nr.2 blz 72.
KALEND	2900	AFAF	00955	16F	A.N. nr.2 blz 93.
BRAILLE	2900	AFAF	00DAF	179	A.N. nr.2 blz 95.
BREUKEN	2900	AFAF	005E1	187	A.N. nr.2 blz 92.
CURSOR2	2900	AFAF	001F2	18D	A.N. nr.2 blz 74.

Schijf nr 2. 85/2

REGRESS	2900	AFAF	0226B	002	A.N. nr.2 blz 76.
PFC-TLK	A000	C2B2	01000	037	A.N. nr.1 blz 65.
PFC-PCH	8200	AFAF	01072	047	A.N. nr.1 blz 65.
PFC-AXX	A000	C2B2	01000	058	A.N. nr.1 blz 65.
GAGSDM	2900	AFAF	00367	068	A.N. nr.2 blz 91.
TEKENEN	2900	AFAF	0058E	06C	A.N. nr.2 blz 90.
BREAKd	2900	AFAF	0053A	072	A.N. nr.2 blz 14.
INFOCO	8200	AFAF	0139D	078	A.N. NR.2 blz. 6.

Aangezien door een misverstand mijnerzijds niet de juiste programmatuur voor de Program Flower Controller op de schijf stond, nu alle drie versie's nog een keer. Excuses.

7 MRT 1985 BRABANT GEBOUW B.R.A.N.
20.30 UUR HEUVEL 7 VORSTENBOSCH.

8 MRT 1985 DEN HAAG EXODUSKERK
20.00 UUR BERESTEINLAAN 263 DEN HAAG.

9 MRT 1985 BELGIE OXACO-CAPENBERGCENTER
10.30 UUR BORSBEEKSE STEENWEG 45. BOECHOUT (BIJ HOVE)

13 MRT 1985 TWENTE MEPA-GEBOUW
20.00 UUR WIERDENSESTRAAT 40 ALMELO.

13 MRT 1985 CENTRUM DE LANTAERN
20.00 UUR UTRECHTSESTRAATWEG 4 NIEUWEGEIN.

14 MRT 1985 NOORD MUNSTERHOES
19.30 UUR MUSTERHEERD. GRONINGEN.

19 MRT 1985 BRABANT DE WERFF
20.30 UUR V/D WERFFSTRAAT EINDHOVEN.

20 MRT 1985 NOORD M van VLISSINGEN
ACHTER DE GROTE KERK 25 LEEUWARDEN.

1 APR 1985 ARNHEM T.O. POSTGIRO
20.00 UUR VELPERWEG 56-A (kelder) ARNHEM

4 APR 1985 BRABANT GEBOUW B.R.A.N.
20.30 UUR HEUVEL 7 VORSTENBOSCH.

10 APR 1985 TWENTE MEPA-GEBOUW
20.00 UUR WIERDENSESTRAAT 40 ALMELO.

11 APR 1985 NOORD MUNSTERHOES
19.30 UUR MUSTERHEERD. GRONINGEN.

11 APR 1985 CENTRUM DE LANTAERN
20.00 UUR UTRECHTSESTRAATWEG 4 NIEUWEGEIN.

13 APR 1985 NOORD-H DE KOPEREN KNOOP
12.00 UUR VAN LIMBURG STIRUM STRAAT AMSTERDAM.

16 APR 1985 BRABANT DE WERFF
20.30 UUR V/D WERFFSTRAAT EINDHOVEN.

17 APR 1985 NOORD M van VLISSINGEN
ACHTER DE GROTE KERK 25 LEEUWARDEN.

20 APR 1985 BELGIE OXACO-CAPENBERGCENTER
10.30 UUR BORSBEEKSE STEENWEG 45. BOECHOUT (BIJ HOVE)

26 APR 1985 DEN HAAG EXODUSKERK
20.00 UUR BERESTEINLAAN 263 DEN HAAG.

Regio's die hier niet bij vermeld staan hebben geen opgaven naar de redactie gestuurd.

INFOCO

*Laden op #8200 (volledig basic).
*de bediening van het programma wijst zichzelf wanneer de volgende tips opgevolgd worden.

-gebruik "HELP" om de commando's teleren kennen.
-vergeet niet om een nieuwe file te openen met "INIT".
-gebruik een simpele druk op "CR" om een invoer te weigeren of om iets ongewijzigd te laten.
-gebruik binnen "FIND" field 0 en record "ALL" om het hele bestand te listen. gebruik "CTRL" om hieruit te ontsnappen.
-"<TEKST>" betekent even geduld.
-"TEKST>" betekent druk op de spatiebalk.
-"TEKST><" betekent geef invoer.
-"TEKST?><" betekent geef "YES of NO.
-een 'piep' is altijd een error.
-alle commando's mogen worden afgekort tot een letter.

*Initialisatie advies:
De fields zijn ingedeeld om een adressenbestand te voeren. Tenzij U de indeling veranderd (zie hieronder) is het misschien slim om ze de volgende namen te geven.

LEDEN

NAAM
STRAAT
CODE
PLAATS
TELEFOON

*Alle voor het aanpassen van het systeem relevante informatie is bij elkaar geplaatst in de regels 2010 t/m 2090.
*regel 2010
-Z: dit is het startadres voor de database.
-U: Dit is het eindadres voor de file. Geheugen boven dit adres wordt door het programma niet gebruikt. 'U' werkt dus als een soort protectie op hoger gelegen data.
Let op. Dit geldt niet als het laden van een file van tape er de oorzaak van is dat 'U' overschreden wordt. Alle andere aktie's houden wel rekening met 'U'. Inclusief het schoonmaken van een file m.b.v. "INIT".
-Y: Dit is het aantal fields dat een record zal bevatten (mag 1 t/m 7 zijn). Aldus geenablede fields moeten geinitialiseerd worden met de bijbehorende regels uit de groep 2020 t/m 2080.
-W: dit is het aantal karakters dat de gebruikte printer op een regel zet.
*Regel 2020 t/m 2080.
-DIM In: dit is de voor de naamstring van het field gereserveerde ruimte. N moet gelijk zijn aan GGI, tenzij GGI>8, want dan moet N gelijk aan 8 zijn.
-GGI: Dit is de lengte in karakters van het betreffende field. hiervoor geldt dat het minstens 1 en maximaal 21 mag zijn.
-Hoewel er niets misgaat als het niet zo is, verdient het met het oog op hardcopy aanbeveling om ervoor te zorgen dat de som van alle GGI's niet groter is dan W-9.
-als GGI=4 gekozen wordt, neemt INFOCO bij het printen aan dat het hiereen numeriek field betreft.
*Regel 2090
Deze regel bevat de gebruikersnaam. Messtal zal hier de naam van een instelling staan. belangrijk is dat de lengte van de string 32 blijft. De string mag alle karakters bevatten met ASCII-waarden tussen #20 en #5F.

```
0REM INFOCO 11184  J.R.MARKS
10P.$12;GOS.i
20aGOS.s
30b$T="COMMAND";D.GOS.g;U.?C<>13
40I.$C="QUIT"O.$C="Q";P.$12;E.
45I.$C="INIT"O.$C="I";G.100
50I.$C="HELP"O.$C="H";G.200
55I.$C="ENTER"O.$C="E";G.300
60I.$C="ALTER"O.$C="A";G.400
65I.$C="DELETE"O.$C="D";G.500
70I.$C="FIND"O.$C="F";G.600
75I.$C="SORT"O.$C="S";G.700
80I.$C="PRINT"O.$C="P";G.800
85I.$C="READ"O.$C="R";G.900
90I.$C="WRITE"O.$C="W";G.1000
95$T="UNKNOWN COMMAND";G.c
100$T="DAY";GOS.g;DD0=V.C
105I.DD0>310.DD0<1;$T="NUMBER 1-31";GOS.e;G.100
110$T="MONTH";GOS.g;DD1=V.C
115I.DD1>120.DD1<1;$T="NUMBER 1-12";GOS.e;G.110
120$T="YEAR";GOS.g;DD2=(V.C)%100
125I.DD2<1;I.?C<>48;$T="NUMBER";GOS.e;G.120
130F.I=1T.1;$T="FILENAME";GOS.g;I.L.C>10;V=10;G.j
140N.;I.?C<>13;$N=C;I.L.N<10;D.$N+L.N=" ";U.L.N>9
145F.I=1T.Y;$T="FIELDNAME";?(I*32+#80C9)=31;GOS.g
150I.?C<>13;$FFI=C;I.L.C>GGI O.L.C>8;G.k
155?(I*32+#80C9)=32;N.;$T="CLEAR FILE?";GOS.f;I.?C=78;G.a
160$T="< WAIT PLEASE";GOS.d;R=0;M=0;F.I=Z T.U-4S.4;!I=#20202020
170N.;$T="FILE EMPTY";GOS.d;LI.P;G.a
200@=12;P.$30''''$D"ENTER  - ADD RECORDS"&Z+X*M+X
210P.$D"ALTER  - EDIT RECORDS"'$D"DELETE - ERASE RECORDS"'
220P.$D"FIND   - SEARCH FIELDS"'$D"SORT   - SORT FIELDS"'
230P.$D"PRINT  - HARDCOPY FILE"'$D"INIT   - OPEN FILE"'
240P.$D"READ   - LOAD FILE"'$D"WRITE  - SAVE FILE"'
250P.$D"QUIT   - STOP"';LI.P;G.a
300R=M+1;A=X*R+Z+X-1;I=?A:#77
310I.A>=U;$T="PROTECTED MEMORY";R=M;G.c
320?A=I;I.?A<>I;$T="NO MEMORY";R=M;G.c
330M=R;A=A-X+1;F.I=1T.Y;$T=FFI;GOS.g;J=L.C
340I.I=1;I.?C=13;I=Y;N.;M=M-1;R=M;G.b
350I.J>GGI;V=GGI;G.j
360D.C?J=32;J=J+1;U.J>GGI;F.J=0T.GGI-1;A?J=C?J;N.;A=A+GGI;N.
370GOS.s;G.300
400I.M=0;$T="NO RECORDS TO ALTER";G.c
410$T="RECORD";GOS.g;I.?C=13;G.b
420R=V.C;I.R<10.R>M;$T="UNKNOWN RECORD";GOS.e;G.410
430GOS.s;A=X*R+Z;F.I=1T.Y;$T=FFI;GOS.g;J=L.C;I.J>GGI;V=GGI;G.j
450I.?C<>13;D.J?C=32;J=J+1;U.J>GGI;F.J=0T.GGI-1;A?J=C?J;N.
460A=A+GGI;N.;GOS.s;G.410
500I.M=0;$T="NO RECORDS TO DELETE";G.c
510$T="RECORD";GOS.g;I.?C=13;G.b
520R=V.C;I.R<10.R>M;$T="UNKNOWN RECORD";GOS.e;G.510
530GOS.s;A=X*R+Z;$T="DELETE?";GOS.f;I.?C=78;G.510
550B=X*M+Z;F.I=A T.B S.4;!I=I!X;N.;M=M-1;I.R>M;R=M
560GOS.s;$T="RECORD DELETED";GOS.d;LI.P;G.510
600I.M=0;$T="NO RECORDS TO FIND";G.c
```

```
605$T="FIELD";GOS.g;F=V.C;I.?C=13;G.b
610I.F=0;I.?C=48;G.650
615I.F<1O.F>Y;$T="NUMBER 0- ";GOS.1;G.605
620$T="STRING";GOS.g;DIMZZ-1;B=0;G=0;I.?C=13;G.605
625A=Z;I.F>1;F.I=1T.F-1;A=A+GGI;N.
630D.B=B+1;A=A+X;H=1;F.J=0T.L.C-1;I.C?J<>A?J;H=0
635N.;I.H;G=G+1;ZZG=B
640U.B=M;I.G=0;$T="NOT FOUND";GOS.d;LI.P;G.620
645F.H=1T.G;R=ZZH;GOS.s;$T="FOUND";GOS.d;LI.P;N.;G.620
650$T="RECORD";GOS.g;I.?C=13;G.605
655I.$C="ALL"O.$C="A";G.670
660R=V.C;I.R<1O.R>M;$T="UNKNOWN RECORD";GOS.e;G.650
665GOS.s;G.650
670$T="LIST";F.R=1T.M;GOS.s;GOS.d
675$128=";D.KEYK;U.K=32O.?#B001&64=0;G.680";!5=128
680I.K<>32;R=M
685N.;R=1;G.650
700I.M<2;$T="NO RECORDS TO SORT";G.c
710$T="FIELD";GOS.g;F=V.C
720I.F<1O.F>Y;$T="NUMBER 1- ";GOS.1;G.710
725$T="    < SORTING";GOS.d;@=3
730D.H=0;S=H;I.F>1;F.I=1T.F-1;S=S+GGI;N.
740F.J=M-1T.1S.-1;A=Z+X*J;G=A+X;F.K=S T.GGF-1+S
750I.A?K>G?K;GOS.q;H=H+1;G.760
755I.A?K=G?K;N.
760N.J;P.$13H;U.H=0;$T="FILE SORTED";GOS.d;LI.P;G.a
765qF.L=0T.X-1;B=A?L;A?L=G?L;G?L=B;N.;R.
800P.$21$2;GOS.880;P.'';F.I=1T.W/2-5;P.$32;N.;P.$N'
810@=0;P.DD0"/"DD1"/"DD2'''"      "
820F.I=1T.Y;P.$FFI;F.J=0T.GGI-L.FFI;P.$32;N.;N.;P.'';A=Z+X
830F.I=1T.M;@=3;P.I" ";@=4;F.J=1T.Y
840I.GGJ=4;F.L=0T.3;L?128=A?L;N.;?132=13;P.V.128;G.860
850F.K=0T.GGJ-1;P.$A?K;N.
860P." ";A=A+GGJ;N.;P.';N.;P.''';GOS.890;P.$3$6;G.b
880GOS.890;F.I=1T.W/2-16;P.$32;N.;P.$E'
890F.I=1T.W;P."-";N.;P.';R.
900$T="< LOADING";GOS.d;P.$13;I=X
910X=128;!X=X+16;X!Z=Z;X?4=X;$X+16=N;LI.#FFE0
920M=!Z;R=1;X=I;!Z=#20202020;G.a
1000$T="   < SAVING";GOS.d;P.$13;!Z=M;I=X
1010X=128;!X=X+16;X!2=Z;X!6=Z;X!8=Z+I+1+M*I;$X+16=N;LI.#FFDD
1020X=I;!Z=#20202020;G.a
2000iDIMDD2,N10,FF7,GG7,T32,C32,E32,O32,D33;R=1;P=#FFE3
2010Z=#2900;U=#8000;Y=5;W=80
2020DIMI8;FF1=I;GG1=16;$I=""
2030DIMI8;FF2=I;GG2=21;$I=""
2040DIMI7;FF3=I;GG3=7;$I=""
2050DIMI8;FF4=I;GG4=15;$I=""
2060DIMI8;FF5=I;GG5=12;$I=""
2070DIMI0;FF6=I;GG6=0;$I=""
2080DIMI0;FF7=I;GG7=0;$I=""
2090$E="  'ACORN COMPUTER CLUB  NOORD'  "
2100X=0;F.I=1T.Y;X=X+GGI;N.;$N="  INFOCO  "
2103M=A.M%1000;I.'Z<>#20202020;M=0
2105F.I=0T.31;O?I=E?I+96;I.E?I>63;O?I=E?I+32
2107N.;O?32=13;I.M=0;R=0
2110F.I=0T.X-1;Z?I=32;N.
```

```
 2120$D="                              *";D?32=11;R.
 3000sP.$30;?#E1=0;P.$O;GOS.u;P."+++++÷÷▬÷+"$N"+↓▬↓↓++"
 3010!#804B=!#804B:#80808080;!#804F=!#804F:#80808080
 3015!#8053=!#8053:#8080;GOS.n;GOS.u
 3020P.$D">"';GOS.u;GOS.r;P.$O;R.
 5000uP."++++++++++++++++++++++++++++++";R.
 6000r@=32;P.R;A=R*X+Z;F.J=1T.Y;P.$D$FFJ;?#E0=10
 6020F.I=0T.GGJ-1;P.$A?I;N.;P.';A=A+GGJ;N.
 6030I.Y<7;F.I=Y T.6;P.$D';N.
 6040R.
 7000nP.$M/100+144$M%100/10+144$M%10+144;R.
 8000dP.$30''''$D$T" >";R.
 9000gGOS.d;P."<";$128=";D.KEYK;U.K;G.z";L=0;D.!5=128
 9010zI.K>31;I.K<>127;I.L+L.T<29;P.$8$K"<";C?L=K;L=L+1
 9020I.K=127;I.L>0;L=L-1;P.$8$K"<  "$8
 9030WAIT;WAIT;WAIT;WAIT;WAIT;WAIT;WAIT;L?C=13;U.K=13;R.
 9040fD.GOS.g;U.$C="YES"O.$C="Y"O.$C="NO"O.$C="N";R.
10000eP.$7;GOS.d;LI.P;R.
10010cGOS.e;G.b
10020kV=8;I.GGI<V;V=GGI
10030jST.V,C;?T=32;S=0;I.V<10;S=1
10040$T+S=C;$T+2=" CHARACTER";I.V>1;$T+L.T="S"
10050GOS.e;I=I-1;N.
10060lT?9=Y+48;G.e
```

N.B.
DOSgebruikers dienen in zowel regel 910 als regel 1010 voor de 'LINK' toe te voegen 'X?23=13;'.
Ook de regels 3000 en 5000 verdienen enige toelichting.
De hier afgedrukte karakters leest U als volgt:
'+' : geinverteerde SPC $128
'÷' : geinverteerde '>' $158
'▬' : geinverteerde '-' $141
'↓' : geinverteerde '<' $156
Nu zult U zeggen dat is nu wel leuk en aardig maar hoe krijg ik die dingen in mijn listing?
Dat is echter tamelijk eenvoudig.
Voer eerst de regels in met op de plaats van de probleemkarakters een '?' of iets dergelijks.
Daarna list U een van de regels en tikt:
P.$128" "$141" "$156" "$158'
Nu loopt u met de [COPY] toets over de regel heen.
Zodra U daarbij een '?' tegenkomt copieert U die niet, maar U verplaatst de cursor naar de plek waar het benodigde karakter op het scherm staat.
Daar drukt U op [COPY] en het enige wat U nu nog rest is terug te gaan naar het volgende karakter in de regel.
Hierna wordt de rest van de regel op de zelfde manier behandeld.

HARDWARE KRONKELS

banksselect 64 K & 16 K kaart met DOS

Allereerst enkele afspraken die noodzakelijk zijn bij deze modificatie.

1- de schakelkaart, de 16 K kaarten en de DOS kaart komen buiten de ATOM kast.
Dit moet gewoon omdat er voor twee eurokaarten gewoon geen plaats is in de kast.

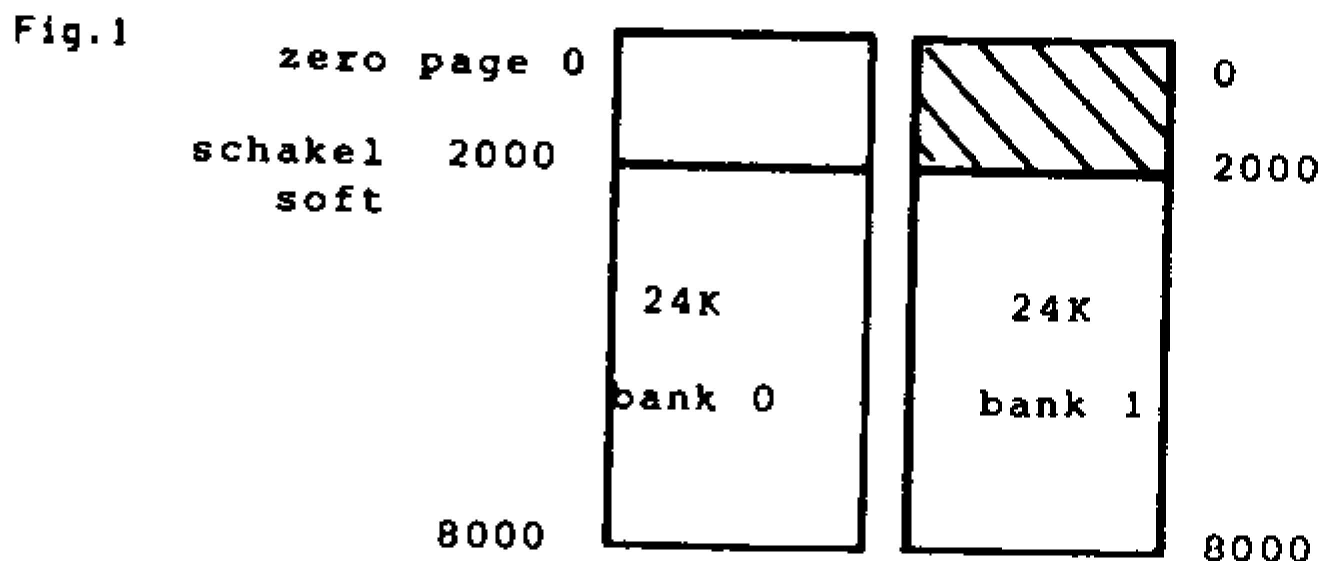
2- Wanneer u een 64 k kaart heeft blijft deze uiteraard in de ATOM-kast.

De in de handel verkrijgbare 64 k kaarten bestaan meestal uit 2banken van 32 k. Dit komt in onze ATOM neer op twee banken van #0000 tot #7FFF. Bij het overschakelen van de ene bank naar de andere moeten we op en aantal punten letten.

a- de zero-page zal moeten blijven bestaan i.v.m. de broodnodige informatie en vectoren.
b- de op #1000 aanwezige schakelsoft moet aanwezig blijven. (het lijkt, gezien het aantal systemen die dit gebied reeds gebruiken, een goede suggeatie dit gebied landelijk daarvoor aan te wijzen)

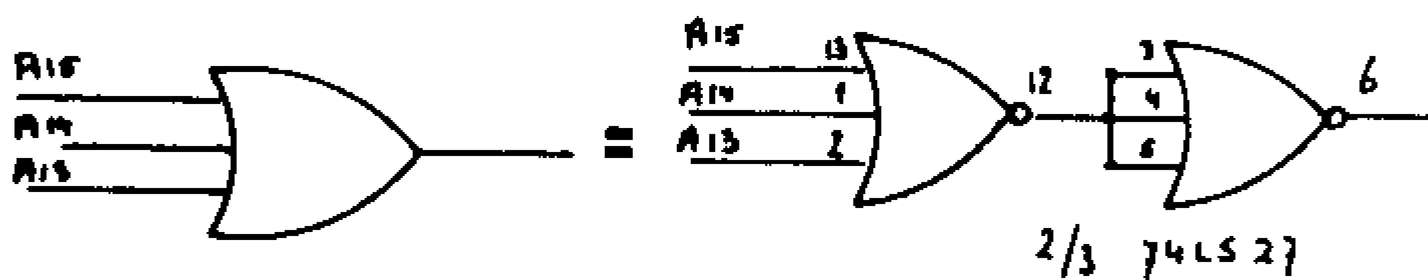
Het nu volgende figuur zal e.e.a. verduidelijken*

Fig.1



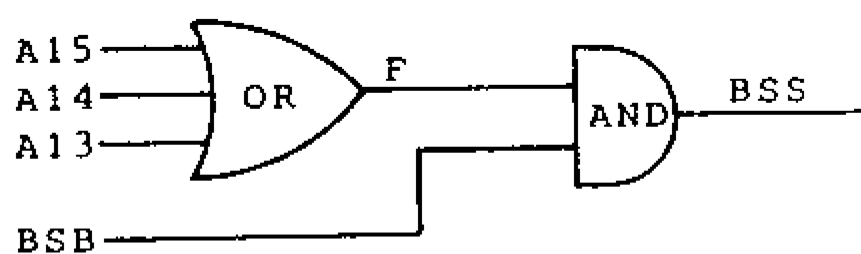
De adressen A15,A14 en A13 bepalen de grens <#2000 of >#2000.
Als een van deze adressen dus actief,hoog wordt(1),is het geadresseerde adres in ieder geval >#2000.
Door nu deze adressen samen te voegen in een OR-poort zal indien een van de adressen hoog wordt de uitgang van de poort ook hoog worden.

Fig.2



Voor het omschakelen van de ene naar de andere bank hebben we een bank schakel bit (BSB) nodig.
We kunnen hier eenvoudig weg een schakelaar voor nemen maar ik heb voor een software matig BSB gekozen. Zodat het dus mogelijk is in een programma om te schakelen om b.v. data uit de andere bank te halen.
Voor BSB heb ik een bit van het schakelbyte #BFFF gekozen en wel het hoogste bit, bit 7. Dit bit werd tot nu toe gebruikt om de gestapelde RAM's op #E000 (#1000) te selecteren.
Door nu het BSB en onze vorige OR-poort te ANDen krijgen we devogende situatie:

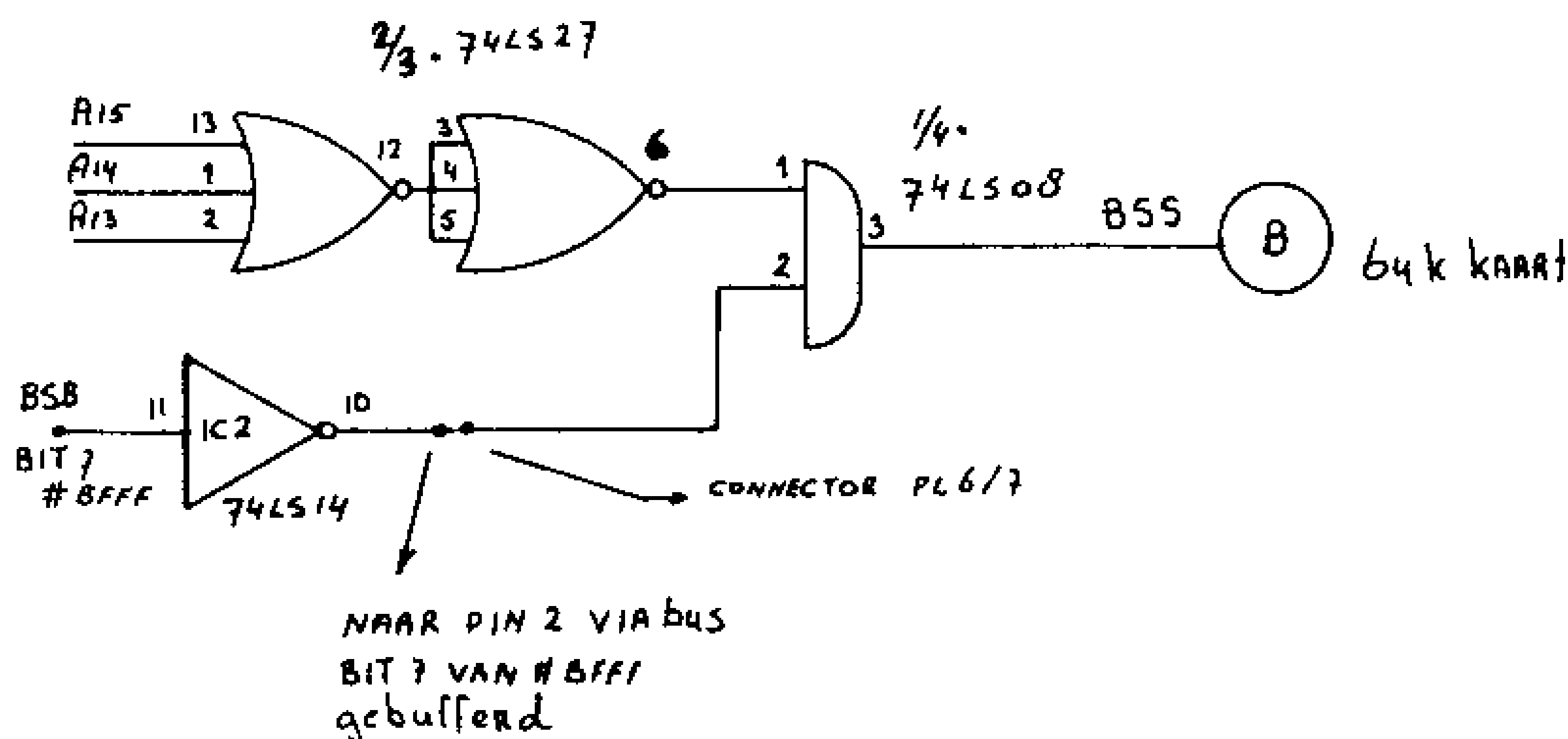
Fig.3



bsb	f	bss
0	0	0
0	1	0
1	0	0
1	1	1

Uit de tabel kunnen we afleiden dat alleen wanneer een van de adressen hoog is en de BSB hoog is het bank select signaal hoog wordt. Dit bank select signaal zullen BSS noemen. Dit BSS verbinden we met de B ingang van de 64 k kaart.
Deze modificatie is dus alleen maar zinvol als u een 64 k kaart heeft en softwarematig wilt schakelen tussen de beide bank's.
Hoe verwezenlijken we dit in de praktijk?
Allereerst iets over het BSB (bit 7 van #BFFF). Dit signaal moet terug van de schakelkaart naar de ATOM-kast. Door de lange afstand tussen de kaart en de kast is het verstandig om dit signaal te bufferen. Dit kan met een nog vrije poort op de schakelkaart, de pennen 10 en 11 van IC.2 (74LS14). Daarna via b.v. pin 2 van de bus, die ook nog vrij is, terug naar de ATOM.
Het uiteindelijke schema met de nodige verbindingen ziet er als volgt uit:

Fig.4

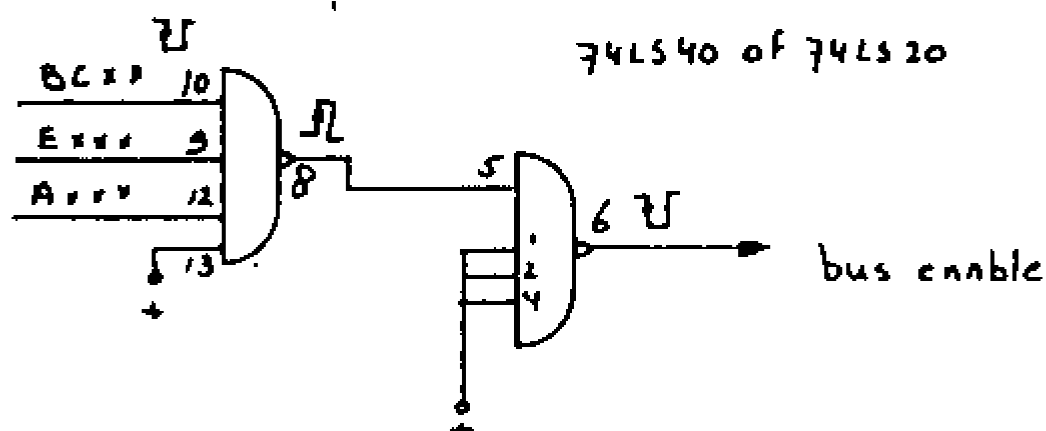


Het nu volgende deel is voor 64 k en 16 k kaart gebruikers van blang als deze het DOS gebruiken.
Enkele belangrijke punten:

1- DOS controller op #0A00 - #0A7F
2- schakelbyte #BFFF
3- toolkits #A000 - #AFFF
4- DOS - ROM #E000 - #EFFF

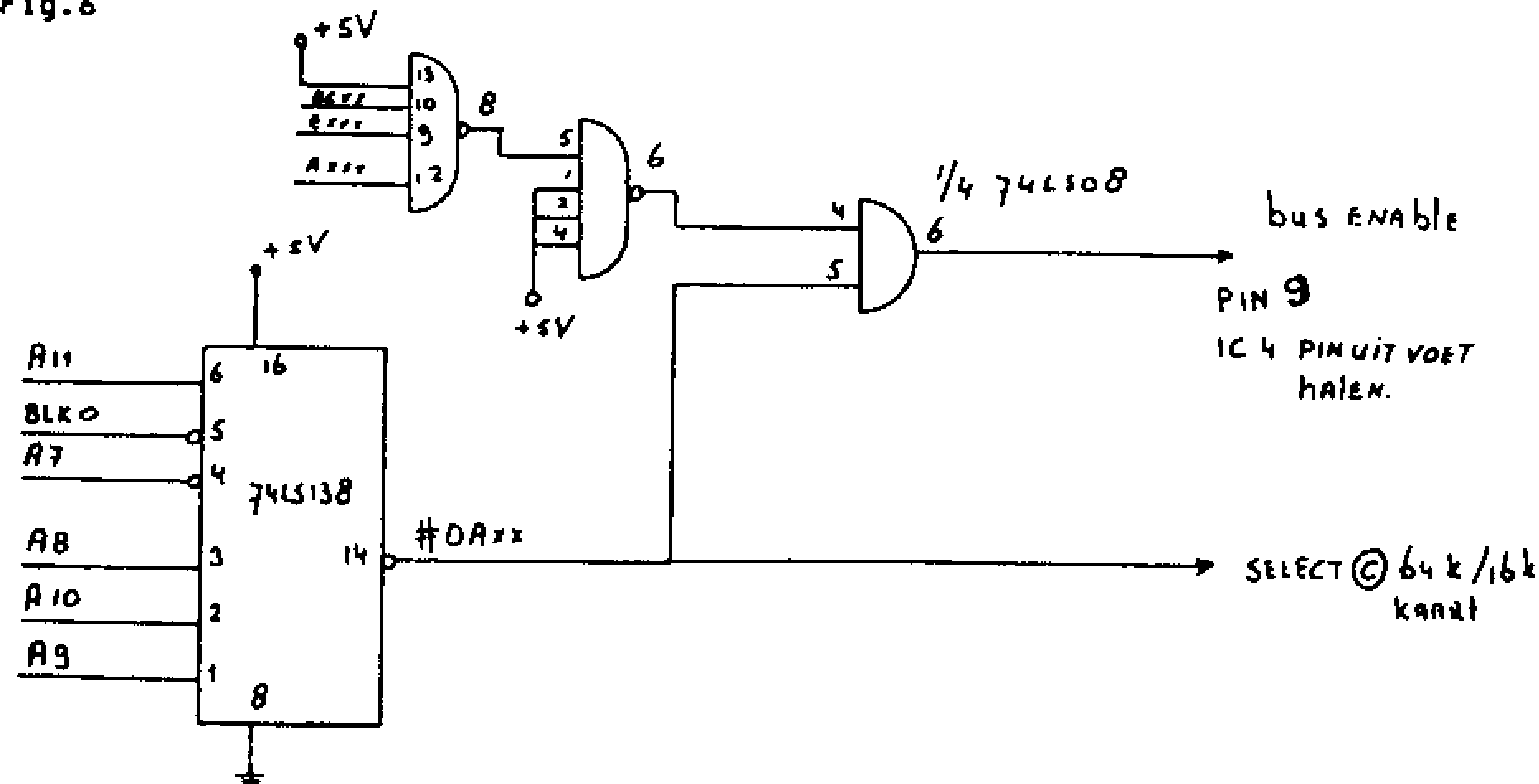
Voor het naar 'buiten" brengen van de adressen onder de punten 2, 3 en 4 is de 'spin' op IC 23 verantwoordelijk (zie schakelkaart documentatie). Met het naar buiten brengen bedoelen we dat de bus - buffer, IC.4, wordt 'aangezet' zodat de data via de buffer naar de buitenwereld kan gaan. de schematische voorstelling van de 'spin' is als volgt:

Fig.5



"Dus niet #BFFF maar #BCXX wordt buiten de bus gebracht omdat #BFFF niet in onze ATOM wordt uitgedecodeerd. Het #XFFF gebied wordt dus op de schakelkaart verder uitgedecodeerd. In een OR-poort uiteindelijk het uiteindelijk #BFFF verkregen.
Dan houden we #0A00 tot #0A7F over.
De bus moet ook hier nu voor worden opengeset doordat we een aantal decoder ic kunnen laten vervallen(hierover straks meer)
Tevens zal het geheugen uitgezet moeten worden i.v.m het zogenaamde "dubbel parkeren".
De decodering voor dit adres gebied zit al op de dos kaart.We kunnen dit echter niet gebruiken omdat dit te ver buiten de kast zit en er timing problemen ontstaan als we dit signaal terug voeren binnen de bus om daarna de bus weer open te zetten.
We zullen dus dezelfde decoder na moeten bouwen alleen dan binnen de kast van de ATOM.
Door daarna het signaal samen met het signaal van de "spin" weer te anden krijgen we het signaal bus enable. Een en ander is is in het volgende figuur getekend.
Dit signaal (bus enable) gaat dan naar pin 9 van IC4 deze pin moet uit de voet gehaald worden.

Fig.6



Het signaal #0AXX gaat tevens nar de C ingang(select) van de 64k kaart zodat deze wordt uitgezet als de doscontroller wordt geactiveerd.
De 16k kaart heeft niet zo'n select ingang dus zullen we die moeten maken.
Allereerst moet zo'n tweede 16k kaart natuurlijk worden aangepast voor het adres gebied van #0-#3FFF zoals al eerder werd beshreven in Acorn Nieuws nr.5 2e jaargang.
Dit staat op de bladzijden 40 t/m 42.
Hoe maken we een select ingang op de 16k kaart?
We zullen dus op een of andere manier het signaal #0AXX ook naar de tweede 16k kaart moeten brengen.
Als we het schema erbij pakken blz.62/63 Acorn Nieuws nr:1 3e jaargang kunnen we het een en ander verduidenlijken.
Verwijder de jumper bij poort N7 waarbij "tijdelijk" staat.
Verbind nu via een vrije connector aansluiting of gewoon met een draadje het signaal #0AXX (pin 14 74LS38 Dos kaart) met poort N7 pinnen 5 en 6.
Er zijn een aantal vrije pennen op de via connector bus nl:
pin 2 (die wordt gereserveerd voor bankselect)
pin 23,pin 24,pin 27 en pin 31
Haal op de 16k kaart de volgende pinnen uit de voeten:

1) pin 4 poort N7 dit is ic 13
2) pin 5 van ic 14 (74LS138)

en leg nu tussen deze twee uitgebogen pootjes een draadje..

Wat hebben we gedaan ?
Wel,als nu het #0AXX signaal komt (low actief) wordt dit door poort N7 geinverteerd (het wordt dus high) en aan de enable ingang van de decoder gezet.

Deze ingang moe low zijn voor het aanzetten, dus zal nu uitgezet worden en het geheugen is gedisabled.
En zie hier is een select ingang ontstaan.
De decodering voot #0Axx en de rest van de poortjes zijn gestapeld aangebracht op een aantal IC's in de buurt van de bus buffers op de print.
Hier zitten namenlijk alle data en adres touwtjes zodat de verbindingen kort gehouden kunnen worden.
IC 5&6 zijn ook uit de print verdwenen (zie artikel over ombouw 16k kaart) zodat ook hier met een tussen voet een tweetal ic's zijn te plaatsen.
Rest mij u nog veel succes te wensen met de nabouw van het een en ander,bij eventuele problemen kunt u gerust bellen.

DE DISKROM

1 — Inleiding

De DISKROM is een 4k-eprom, die een aantal handige uitbreidingen bevat om het diskgebruik te vergemakkelijken. De bedoeling is, dat de diskrom geplaatst wordt in een van de voetjes in de schakelkaart. Voor diegenen die geen schakelkaart bezitten, maar wel voldoende geheugenruimte is het ook mogelijk om van de diskuitbreidingen gebruik te maken door middel van de speciaal voor dit doel aangepaste source (zie punt 3).
De meeste routines in de diskrom zijn geschreven door ondergetekende, echter bekende utilitie's als Format, Infall en Compact zijn enigszins in gewijzigde vorm overgenomen. Hierbij bedank ik iedereen die mij bij het samenstellen van deze rom van advies gediend heeft, met name Henk Reinders en Henk van der Heijden.
De diskrom is onder te verdelen in twee delen:

1. Een uitbreiding van de BASIC-statements
2. Een uitbreiding van de command line statements (*)

Na veel wikken en wegen is voor deze kombinatie gekozen. Er zijn namelijk een aantal voordelen aan verbonden als men de Command Line Interpreter (CLI) gebruikt om statements af te werken. Om te beginnen is de herkenbaarheid groot: men weet direkt dat het kommando betrekking heeft op het Disk Operating System. Een ander voordeel is dat de voorhanden zijnde DOS-routines kunnen worden gebruikt, dit in tegenstelling tot BASIC-statements. Om een voorbeeld te noemen: bij BASIC wordt altijd geinterpreteerd met LDA (#05),Y en in de CLI met LDA #100,Y. Als er nu een '*' voor het kommando staat kan men dus ook die DOS-routines gebruiken die interpreteren met LDA #100,Y. Een ander voordeel is dat er met dezelfde syntax als in de standaard Acorn DOS-rom gewerkt kan worden, hetgeen het gebruik overzichtelijker maakt.
Natuurlijk zijn er ook nadelen aan het werken met een tweede CLI interpreter verbonden. Ten eerste zal altijd de CLI-vector (#206,#207) veranderd moeten worden. Normaal staat deze op adres #E3E5 (in de DOS-rom). Deze nu zal naar het beginadres van de interpreter van de diskrom moeten wijzen. Het gebruik in een schakelkaart vereist dat de goede box voor moet staan. Daarom is in de diskrom een routine opgenomen die zichzelf kopieert naar RAM en vandaar de goede box opzoekt bij het binnenkomen van de CLI. Bij het verlaten van de diskrom wordt de oorspronkelijke box weer voorgezet: het is dus mogelijk om vanuit de editor de standaard DOS-routines uit te voeren. Zie voor een gedetailleerde beschrijving punt 3.
Over de namen van de statements het volgende. Er is gekozen voor namen, zoals die in de BBC-computer voorkomen. Dit is gedaan om standaardisatie in Acornformaat te kontinueren. Statements die in de Disk Filing System (DFS) van de BBC voorkomen zijn:
*ACCESS, *BACKUP, *BUILD, *COMPACT, *COPY, *DESTROY, *DUMP, *ENABLE, *HELP, *LIST, *RENAME, *TYPE, *WIPE.
Ook het BASIC-statement "CHAIN" behoort tot de BBC-BASIC.

2 — De uitbreidingen.

2.1 De BASIC-statements

CHAIN "filenaam" CH.
Met de opdracht CHAIN "filenaam" kan er vanuit een BASIC-programma een ander

BASIC-programma geladen worden in dezelfde tekstpagina en daarna meteen gerund. Voordat het programma gerund wordt, worden eerst de dimpointer (#23,#24) en TOP (#0D,#0E) goed gezet. Net als bij FIN en FOUT is het mogelijk om de filenaam in een string te zetten. In tegenstelling tot LOAD $A wordt CHAIN $A wel uitgevoerd. Met name bij het maken van een serie programma's die vanuit een hoofdprogramma opgeroepen worden, is dit statement erg handig. CHAIN werkt ook met het Cassette Operating System.

DISK DI.

Met dit statement wordt eerst het DOS en tevens de CLI-vector geinitialiseerd voor het gebruik van de extra statements. Deze routine moet alleen gebruikt worden door diegenen, die geen bootstrap gemaakt hebben of een aangepaste resetroutine (BREAK) gebruiken. Zie voor verdere informatie punt 3.

FCOS FC.

Dit BASIC statement is bekend uit de Josbox en zet alle poorten en vectoren goed om een file te laden of te saven met 1200 baud op de cassette. Nadat het laden of saven beeindigd is, wordt het DOS en de diskbox weer geinitialiseerd.
De wijzigingen zoals vermeld in Acorn Nieuws nr.3-5 zijn in de FCOS-routine verwerkt.

2.2 CLI commando's:

*ACCESS (<L>) (<drive>) *A.

Met *ACCESS (<drive>) worden alle files op de diskette "geunlocked" en met *ACCESS L (<drive>) worden alle files "gelocked". In wezen wordt hetzelfde uitgevoerd als *(UN)LOCK <filenaam> uit de DOS-rom.

*BACKUP (<drive> <drive>) (<beginadres> <eindadres>) *BA.

Om *BACKUP te gebruiken moet er eerst een *ENABLE gegeven worden.
Met *BACKUP <return> wordt een hele schijf gekopieerd sector bij sector. De defaultwaarden zijn zoals ze bij alleen een <return> gelden: kopieren van drive 0 naar drive 0 in het buffergebied #2400 tot #6700. Dit betekent dat de schijven zes keer verwisseld moeten worden. Het is ook mogelijk om middels de instruktie *BACKUP 0 0 2400 7400 in vijf keer wisselen een disk te kopieren. Immers er worden dan 80 sectoren tegelijk gekopieerd. Voor diskgebruikers met twee of meer drives is het natuurlijk ook mogelijk om te kopieren van drive 0 naar drive 1 of omgekeerd, of van drive 1 naar drive 3 etc. Er kan dan bijv. de instructie *BACKUP 0 1 gegeven worden.Als er na het opgeven van de drives geen <return> gegeven wordt, moet het buffergebied opgegeven worden met bijvoorbeeld: *BACKUP 0 1 2300 4000. Diegenen die geen 16K-kaart bezitten kunnen dit programma dus ook gebruiken door het buffergebied #2300-#4000 te kiezen. Er moet dan wel 15 (!!) keer van schijf verwisseld worden bij het ontbreken van een tweede drive.
Samenvattend zijn er dus de volgende mogelijkheden:
*BACKUP
kopieert in drive 0 in het gebied 2400-6700
*BACKUP <drive> <drive>
kopieert van ene drive naar andere in buffergebied 2400-6700
*BACKUP <drive> <drive> <buffergebied>
kopieert van de opgegeven drives binnen de gegeven buffer

*BUILD <filenaam> *B.

Met het commando *BUILD <filenaam> kan er een tekstfile gekreeerd worden, die afgesloten dient te worden met een control-D of met het ASCII-karakter #7D (een geinverteerde teksthaak), gevolgd door een return. Met *EXEC <filenaam> kan de file geexecuteerd worden, waarbij de computer de verschillende opdrachten

uitvoert. Met *TYPE <filenaam> kan de file gelezen worden.

***COMPACT (<drive>)** ***C.**

Met het statement *COMPACT (<drive>) worden de "gaten" op de diskette opgevuld. Alle files op de schijf worden netjes achter elkaar op de schijf gezet. Vooral op schijven waar veel files weggehaald zijn, worden niet altijd alle sectoren gebruikt voor data-opslag. Alle files worden geladen in het geheugengebied #2C00-#3BFF

***COPY (<drive> <drive>) <filenaam>** ***COP.**

Met *COPY <drive> <drive> <filenaam> wordt er een file gekopieerd. Indien de drivenummers gelijk zijn wordt er gewacht totdat de schijven verwisseld zijn. De file wordt geladen vanaf #2900 maar alle filespecifikaties worden uiteraard meegekopieerd. Ook tekstfiles, eproms en random access files kunnen gekopieerd worden.
Als er *COPY <drive> <drive> <return> ingetikt wordt, kunnen alle files op een bepaalde schijf gekopieerd worden. Bij iedere file wordt er gevraagd of deze gekopieerd moet worden. Uiteraard wordt als er hetzelfde drivenummer gegeven wordt, gevraagt om de schijven te verwisselen.Met *COPY <return> wordt er gekopieerd in drive 0. Dit is hetzelfde als *COPY 0 0.
Het programma kan op ieder gewenst moment onderbroken worden met de escape-toets. Files die met *COPY gekopieerd zijn, zijn nooit gelocked.

***DCAT (<drive>)** ***.**

*DCAT (<drive>) geeft een driekoloms katalogus met aanvullende informatie over het drivenummer, de qualifier, het aantal programma's, de option voor de bootfile en het aantal vrije sectoren. Als een schijf vol is - 31 files of alle sectoren gebruikt - wordt dit aangegeven met de mededeling "DISK FULL".

***DESTROY (<drive>)** ***DE.**

Om *DESTROY te gebruiken moet er eerst een *ENABLE gegeven worden.
Met *DESTROY (<drive>) kunnen alle programma's verwijderd worden. Uiteraard blijft de data op schijf staan, maar in de catalogue staan geen programma's meer: de schijf is leeg. Enige voorzichtigheid is hierbij geboden. Mocht men achteraf spijt hebben van de gedane opdracht dan is dit te herstellen met een peek en poke-grapje.
Met een hexdump wordt het aantal programma's geteld vanaf #2008 na een *(D)CAT gegeven te hebben. Dit getal wordt vermenigvuldigd met 8 en in #2105 gezet. Dan tikt men in:

```
?#2105=prog.*8
LINK #E75B (zet motor aan)
LINK #E74A (save catalogue)
```

en het geheel is weer in de oude staat gebracht.

***DUMP <filenaam>** ***DU.**

Met *DUMP <filenaam> wordt er van een file een ASCII en hexadecimale dump gemaakt. Zie verder bij *LIST en *TYPE.

***ENABLE** ***E.**

Met *ENABLE worden de kommando's *BACKUP, *DESTROY en *FORMAT toegankelijk gemaakt voor de diskgebruiker. Dit om rampzalige gevolgen van tikfouten te voorkomen.

***FORMAT (<drive>)** ***F.**

Om *FORMAT (<drive>) te gebruiken moet er eerst een *ENABLE gegeven worden.
Formatteerprogramma voor~-*!1 40-track~-*!0 diskettes.

***HELP** ***H.**

Met *HELP worden alle standaard DOS-kommando's en de extensions uit de diskrom

uitgeprint.

***INFALL (<drive>)** ***I.**

*INFALL (<drive>) geeft informatie over alle files op een disk omtrent he executie-adres, beginadres, lengte van de file en de beginsector van de fil (*INFO).

***LIST <filenaam>** ***LI.**

Met *LIST <filenaam> kan er een file, dat uiteraard een BASIC-programma moe zijn, uitgeprint worden op het beeldscherm, zonder dat het geladen wordt in he tekstgeheugen. Als de file geen BASIC-programma is verschijnt er natuurli onzin. Zie verder bij *DUMP en *TYPE.

***OPT (<option>) (<drive>)** ***O.**

Met *OPT (<option>) (<drive>) kan de option van de bootfile, die op de sc staat veranderd worden:
*OPT 0 doet niets
*OPT 1 laad de bootfile (*LOAD !BOOT)
*OPT 2 runt de bootfile (*RUN !BOOT)
*OPT 3 executeert de bootfile (*EXEC !BOOT)
Zie ook punt 4 over de bootfile.

***QUAL <filenaam>** ***..**

Met *QUAL <filenaam> (<qualifier>) wordt een file in een andere library gezet. Als er na de filenaam een return gegeven wordt zal de qualifier van de file op #20 (default) gezet worden.Ook gelockte files en files in een andere library kunnen veranderd worden. Het idee achter deze "syntaxfout" is om files die per ongeluk in een verkeerde library terecht zijn gekomen gemakkelijk te kunnen veranderen. Na gebruik van het statement is de file nooit gelocked.

***RENAME <oude filenaam> <nieuwe filenaam>** ***RE.**

Met *RENAME <oude naam> <nieuwe naam> kan men de filenaam veranderen. Let hierbij wel op dat de naam niet veranderd wordt in een die reeds voorkomt in dezelfde directory. Gelockte files kunnen niet veran r worden.

*SECTOR (<drive>) *SE.

Dit is een sector-lees programma. Men kan door eenvoudige toetsbediening sectoren lezen en indien gewenst kopieren naar het werkgeheugen. Het programma is "menu-driven". Het menu waar het programma mee begint, ziet er als volgt uit:

- A kopieert sector naar geheugen en maakt een ASCII-dump
- H kopieert sector naar geheugen en maakt een hexadecimale dump
- C kopieert sector naar geheugen en verhoogt het geheugen met een pagina
- P laadt de vorige sector
- M de plaats in het geheugen waarheen de sector gekopieerd wordt, kan veranderd worden
- S de sector, die gekopieerd wordt kan veranderd worden
- Q terug naar BASIC

Als gekozen wordt voor A, H of C wordt telkens automatisch de volgende sector in het buffergeheugen geladen. Het programma begint sector 0 te kopieren op adres #2300.

***TYPE <filenaam>** ***T.**

Met *TYPE <filenaam> kan een tekstfile, die gemaakt is met *BUILD of *SPOOL uitgeprint worden zonder dat er executie van de tekst gedaan wordt, zoals bij *EXEC. Als de betreffende file geen tekstfile is dan verschijnt er natuurlijk

onzin op het beeldscherm. Zie verder bij *DUMP en *LIST.

***VERIFY (<drive>)** *V.

Testprogramma voor diskettes met 40 tracks.

***WIPE (<drive>)** *W.

Met *WIPE (<drive>) kunnen alle files, die onder de bestaande qualifier staan en niet gelocked zijn, verwijderd worden (*DELETE). Bij iedere file wordt er gevraagd of deze verwijderd moet worden. Met de escape-toets kan het programma onderbroken worden.

Indien men over meerdere drives beschikt kan men de betreffende statements ook aanroepen met een ander drivenummer dan 0:
Als U achter de betreffende opdrachten geen getal zet, zal het statement dit beschouwen als zijnde de drive die op dat moment aanstaat.

Samenvatting van de CLI-uitbreidingen

1. Krachtige kopieerroutines met *COPY en *BACKUP.
2. Utilities om sectoren en files te lezen met *DUMP, *LIST, *SECTOR en *TYPE.
3. Verwijderen van files en beveiliging van dem catalogue met *ACCESS, *DESTROY, *ENABLE en *WIPE.
4. Het veranderen van files met *RENAME en *QUAL.
5. Bekende utilities als *COMPACT, *FORMAT, *INFALL en *VERIFY.
6. Het kreeren van tekstfiles met *BUILD.
7. Een uitgebreide *CAT en een auto-DOS-start met *OPT.
8. Een snelle 1200 baud COS-routine.
9. Mogelijkheid tot het schakelen naar een ander BASICprogramma met CHAIN.

3 - De initialisatie

3.1 De diskrom in de schakelkaart

Om de diskrom naar behoren te laten werken, dienen er een aantal voorzieningen getroffen te worden. Ten aanzien van de BASIC-statements is dit niet nodig. Deze doen gewoon mee in ieder schakelsysteem. Voor de CLI-kommando's moet in de eerste plaats de CLI-vektor verzet worden. Deze vektor #206,#207 staat normaal op adres #E3E5, het begin van de interpreter van de DOS-rom. Deze vektor wordt verplaatst naar een adres waar een routine staat, die de diskrom opzoekt op de schakelkaart en voorschakelt. In wezen moeten de extra CLI-kommando's ingeschakeld worden. Dit kan op twee manieren gebeuren:
1. Met het statement DISK,
welke ingebouwd is als BASIC statement in de diskrom. Met DISK wordt er een zoekroutine gekopieerd naar #9FA0 en de CLI-vektor wordt ook op #9FA0 gezet. Dit statement is speciaal ingebouwd voor diegenen, die niet met een bootstrap of aangepaste resetroutine (BREAK) werken.

2. Met een aangepaste resetroutine
In deze routine wordt vaak eerst enkele poorten goed gezet, dan de vektoren en de diskgebruikers zullen dan ook met JSR #E000 de DOS aanzetten. Na het aanzetten van de DOS moet men de CLI-vektor wijzigen in het beginadres van de zoekroutine voor de diskrom. Deze zoekroutine kan natuurlijk overal in het geheugen staan.

In de diskrom zijn nog een aantal veiligheden ingebouwd, die er voor zorgen dat deze rom kan samenwerken met de andere boxen op de schakelkaart, met name in BASIC-programma's. Immers, zoals bekend verandert de P-Charme interpreter de breakvektor in BASIC-programma's.

In de zoekroutine wordt de breakvektor (#202,#203) en het nummer van de box die voorstond, voordat de CLI aangeroepen werd, gesaved op de adressen #9FF0-#9FF2. Als een CLI-kommaando juist is afgewerkt of niet gevonden is wordt er gesprongen naar een subroutine, die de originele breakvektor hersteld en de vorige eprom weer voorschakelt. Hierbij wordt er een klein stukje assembler gekopieerd naar #9FE0-#9FE6.
Als de schrijfroutine-vektoren (#208,#209) veranderd zijn in een adres dat in blok #A000 staat, zoals het geval is met EDIT en CALC wordt er direkt doorgesprongen naar de interpreter van de DOS-rom (#E3E5). Het zal duidelijk zijn dat de extra CLI-kommando's waarbij tekst wordt afgedrukt, niet kunnen samenwerken met bijv. de editor of met GRMOD uit de Josbox.
Veel diskgebruikers zullen de bestaande CLI-kommando's afkorten. Veel voorkomende afkortingen zijn *L. (*LOAD), *S. (*SAVE) en *R. (*RUN). In de interpretertabel van de diskrom zijn de genoemde kommando's ook opgenomen, om de originele afkortingen te kunnen handhaven. Wanneer dit niet gedaan zou zijn, zouden de genoemde afkortingen respektievelijk *LIST, *SECTOR en *RENAME betekenen. Zie voor alle afkortingen punt 5.

Het statement "DISK"

```
:test  JSR #C4E4        \test einde statement
       JSR #E000        \initialiseer DOS

                        \copieer zoekroutine naar #9FA0
       LDX @(eind-begin)
:copy  LDA eind,X
       STA #9FA0,X
       DEX
       BPL copy
                        \zet command-line vector op #9FA0
       LDA #9F
       STA #207
       LDA @#A0
       STA #206
       JMP #C55B        \einde kopieerroutine
```

Het volgende stukje machinetaal wordt door het statement DISK gekopieerd naar #9FA0. Dit gedeelte zou in een bootstrap kunnen of in een aangepaste breakroutine na JSR #E000.

```
:begin
       LDA #9FFF        \kijk welke box aanstaat
       STA #9FF2        \save boxnummer
                        \save breakvector
       LDA #202
       STA #9FF0
       LDA #203
       STA #9FF1
                        \zet breakvector goed
       LDA @#C9
       STA #203
       LDA @#DB
       STA #202
                        \zoek diskrom op
       LDX@#8           \initialiseer teller
:box   DEX
       STX #BFFF
       LDY @#03         \ga zoeken of er DISK staat op adres #A03F
:find  LDA #A03F,Y
```

```
        CMP string,Y
        BNE box
        DEY
        BPL find        \box gevonden
        STX #9FFF
        JMP #A047       \spring naar interpreter

:string
        $P="DISK";P=P+L.P
:eind
```

3.2 De diskutilitie's in het ram-geheugen

Voor diegenen, die niet in het bezit zijn van een schakelkaart, is een speciale source ontworpen, zodat zij ook gebruik van de diskuitbreidingen kunnen maken. Deze source bevat uitsluitend de CLI-kommando's en niet de BASIC-statements FCOS en CHAIN. De voorzieningen zoals vemeld in punt 3.1 zijn niet nodig, omdat er niet geschakeld hoeft te worden in een schakelkaart en zijn derhalve weggelaten. De interpreter is wat aangepast om gebruik in het lagere geheugen (onder de #8000) ook mogelijk te maken. Het assembleerprogramma gebruikt geheugen van #2900 tot ongeveer #7000. De objektcode zal dus boven de #7000 of lager dan #2800 kunnen staan. Om de uitbreiding te initialiseren is alleen een verandering van de CLI-vektor nodig. Het programma geeft dit ook aan. De lengte van de objektcode is ruim 13 pagina's (3,4 kb).

4 — De bootfile

In een DOS-systeem als die van de Atom mag eigenlijk een DOS-autostart niet ontbreken. Over dit is al het een en ander gezegd, o.a. in Acorn Nieuws. Kort samengevat komt het op het volgende neer.
In de resetroutine (BREAK) wordt een test ingebouwd of de Control-toets, of zoals men wilt de Shift-toets wordt ingedrukt. Deze test gebeurt met LDA #B001 en wordt vervolgd met CMP @#BF voor de control-toets of CMP @#7F voor de shift-toets. Als dit waar is wordt er gesprongen naar een routine, die test of de option van de schijf ongelijk aan nul is. Als de option 1 is wordt de bootfile geladen (*LOAD), als de option 2 is geladen en gerund (*RUN) en als de option 3 is geexecuteerd (*EXEC).Het programma !BOOT wordt de bootfile genoemd en moet aanwezig zijn op de schijf die op dat moment in drive 0 zit onder de default qualifier. Zo'n programma kan bijvoorbeeld een ander (hoofd)programma gaan laden en runnen met bijvoorbeeld:

```
10 REM !BOOT
20 CHAIN "NONSENS"
30 END
```

(deze laatste regel is niet noodzakelijk, zoals de oplettende lezer opgemerkt zal hebben)
Bovenstaand programma moet dan wel met BSAVE weggeschreven zijn; de option is 2 (*RUN).

Een andere mogelijkheid is om met *BUILD de volgende tekstfile te kreeren:

```
CHAIN "NONSENS"
```

en deze als bootfile te executeren met option 3 (*EXEC).

Een andere manier is om alle files - dit zijn meestal BASIC of machinetaalprogramma's - af te drukken in een soort menu. Vanuit dit menu kan

dan een programma gekozen worden. De bootfile zal dan het programma laden en gaan runnen.
Onderstaand programma is zo'n bootfile, welke bovendien nog de eigenschap heeft om random access files - weet U wel die beginnen op adres #0000 en gemaakt worden met FIN, FOUT, *SPOOL en *BUILD - niet in het menu te zetten. Bovendien worden die files, van welke het startadres groter is dan #A000 (meestal roms) en kleiner dan #2000 (write protect) ook niet in het menu opgenomen. Ook de bootfile zelf ontbreekt natuurlijk in het menu.
Het zal eenieder duidelijk zijn dat met name de BASIC-programma's, die in zo'n menu voorkomen goed weggezet moeten zijn op schijf. Aan te raden is dan ook om alle BASIC-programma's met het statement "BSAVE" uit de P-CHARME weg te schrijven.

Dan volgt nu de listing van de bootfile:

```
 10 REM !BOOT
 20 DIM PP(31)
 30 FOR N=0 TO 31
 40   DIM P(7);PP(N)=P
 50 NEXT
 60 *NOMON
 70 A=?#2105/8;B=1
 80 FOR N=1 TO A
 90   ?(#2007+N*B)=#D
100 NEXT
110 FOR N=1 TO A
120   P=#2000+N*B
130   IF $P="!BOOT   ";GOTO 190
140   S=!(#2100+N*8)&#FFFF
150   XIF S>#21FF AND S<#9E00
160     $PP(B)=$P
170     B=B+1
180   ELSE
190 NEXT;B=B-1
200 PRINT $12"DISK: "
210 FOR J=0 TO 7
220   PRINT $(?(#2000+J))
230 NEXT
240 FOR J=0 TO 4
250   PRINT $(?(#2100+J))
260 NEXT
270 PRINT '"--------------------------------"'
280 C=B/2+B%2
290 FOR N=1 TO C
300   PRINT $(CH"@"+N),". "$PPN"       "
310   XIF N+C<=B
320     PRINT $(CH"@"+N+C),". "$PP(N+C)'
330   ELSE P.'
340 NEXT
350 PRINT "--------------------------------"'
360 PRINT "DRUK OP EEN TOETS "
370 INKEY I
380 IF I<CH"A" OR I>(CH"@"+B);BEEP 20,30;GOTO 370
390 I=I-CH"@"
400 PRINT '"KEUZE : "$PP(I)
410 P=TOP-17;N=0
420 $P=$PPI;P?7=CH""""
430 DO
440   N=N+1
```

```
450   P=#2000+N*8
460 UNTIL $P=$PPI
470 P=TOP-27
480 Q=#2007+N*8
490 *DIR
500 IF ?Q>#80;?Q=?Q-#80
510 ?P=?Q
520 BEEP 20,30
530 ?#12=?(#2101+N*8)
540 *SETx
550 *RUN"xxxxxxx"
560 END
```

Geheugengebruik

De volgende geheugengebieden worden gebruikt:

1. Zero-page adressen #90 tot #95 voor tijdelijk gebruik.
2. DISK: de adressen #9FA0 tot #9FDB. Deze adressen mogen niet voor andere doeleinden gebruikt worden. Alleen als men gebruikt maakt van een bootstrap of gewijzigde resetroutine (zie punt 3) kunnen deze adressen wel gebruikt worden, want dan is het statement DISK niet meer nodig.
   De breakvektor en het boxnummer worden weggezet op de adressen #9FF0 tot #9FF2 (tijdelijk gebruik).
3. De adressen #9FE0 t/m #9FE5 voor de terugspringroutine (tijdelijk gebruik).

5 De resetroutine (BREAK)

In de onderstaande routine worden eerst de poorten en vectoren goed gezet. Hierna wordt - dit is een persoonlijke initialisatie - de P-Charme voorgezet en dan wordt er gekeken of het DOS aanstaat. Is dit het geval dan wordt deze geinitialiseerd. Tevens wordt de diskrom ingeschakeld door het goed zetten van de CLI-vector. Dan wordt er getest of de control-toets ingedrukt wordt. Wanneer dit inderdaad gebeurt wordt catalogue van drive 0 geladen en gekeken welke option uitgevoerd moet worden ten aanzien van de bootfile. Het gedeelte vanaf :dos is belangrijk voor de bootfile en de diskrom.

```
:begin    LDA @#A;STA #FE                    zet testbyte printer
          LDA @#8A;STA #B003                 zet poorten goed
          LDA @#07;STA #B002
          STA #9FFF;STA #BFFF                zet P-Charme voor
          LDX @#03
:loop     LDA tabel1,X;STA #208,X            zet write-readvectoren
          DEX;BPL loop
          LDA @#AB;STA #203                   zet breakvector
          LDA @#D5;STA #202
          LDA @#00;STA #3EE           )
          LDA @#11;STA #3EF           )      init uitbreidingen
          LDA @#B3;STA #3FB           )
          TXS;INX;STX #E7
          STX #EF;CLI                        *MON
          LDA @#80;STA #E1
          JSR #FD8D
          JSR #F7D1                          print het volgende
          $P="  ACORN ATOM  DOS    "
          ?P=#06;P?1=#0C
          P?12=#0A;P?13=#0D
          P=P+L.P+5;!P=#A0A0D07;P=P+4

:dos      LDA #E000;CMP @#A9                 test of DOS aanstaat
```

```
         BNE einde;JSR #E000              initialiseer DOS
         LDA @disk%256;STA#206         )  zet CLI-vector goed
         LDA @disk/256;STA#207         )  voor de DISKROM
         LDA #B001;CMP @#BF               test op control-toets
         BNE einde
         JSR boot
:einde   JMP #C2CF

:tabel1  !P=#FE94FE52;P=P+4               read-write adressen

:boot    LDA @#29;STA #12
         JSR #E223;LDY @#05               laad catalogue
:label   LDA tabel2,Y;STA #100,Y          zet !BOOT op #100,X
         DEY;BPL label
         STY #EF;INY                      *NOMON
         LDX @#00
         LDA #2106;JSR #E0FB              welke option
         BEQ nothing
         CMP @#01;BNE run
         JMP #E465                        *LOAD bootfile
:run     CMP @#02;BNE exec
         JMP #E50A                        *RUN  bootfile
:exec    CMP @#03;BNE nothing
         JMP #E519                        *EXEC bootfile
:nothing RTS

:tabel2  $P="!BOOT "
         P=L.P+P;?(P-1)=#0D

:disk    LDA #9FFF;STA #9FF2              zie punt 3
         LDA #202;STA #9FF0
         LDA #203;STA #9FF1
         LDA @#C9;STA #202
         LDA @#D8;STA #203
         LDX @#08
:box     DEX;STX #BFFF
         LDY @#03
:find    LDA #A03F,Y;CMP string,Y
         BNE box;DEY;BPL find
         STX #9FFF;JMP #A047
:string  $P="DISK";P=P+L.P
```

6-De schijf

Op de schijf van de diskrom staan de volgende files:

- DBOX-V1 - De diskrom voor op #A000.
- DISK-V1 - De source van de diskrom
- DSRC-V1 - De source van de diskutilitie's voor diegenen, die geen schakelkaart hebben.
- !BOOT - De in punt 4 beschreven bootfile.
- BREAKd - Een resetroutine die eventueel naar eigen inzicht aangepast kan worden.

GAGS SPELLETJE BURGERTIME

Het spel is gemaakt voor de GAGS-box en maakt gebruik van de geheugenkaart. Het doel van het spel is om zoveel mogelijk hamburgers te maken. Deze kunnen gemaakt worden door over het BOVENSTE broodje te lopen.
Voor elk gevallen deel krijgt men 10 punten, 50 EXTRA punten krijgt men voor het pletten van de UI tussen de vallende delen. Uw enigste verdediging is weglopen of het strooien van PEPER welke LINKS- en RECHTSboven te verkrijgen zijn, het strooien gebeurt door op ACTION te drukken. Wanneer de UI de peperwolk tegenkomt is hij tijdelijk uitgeschakelt.
BURGERTIME bevat 6 LEVELS welke elk 2 keer doorlopen moeten worden. Wanneer dit gebeurt is, of men is 3 mannetjes kwijt krijgt men de SCORE te zien.

VEEL SUCCES MET B U R G E R T I M E !!!!.

```
 10REMburgertime van Kees van Oss
 20 BASE#44;REM(of#98)
 25 NOSNOW
 30 F.I=0TO6S.3
 40 CR.BU1,0,0,0,#3F,#1F,#1F,#F,7,0,0,0,#FC,#F8,#F8,#F0,#E0/A:I;N.
 50 F.I=1TO7S.3
 60 CR.ME1,0,0,0,0,#2A,#55,#2A,#15,0,0,0,0,#A8,#54,#AA,#54/A:I;N.
 70 F.I=2TO8S.3
 80 CR.SL1,0,0,0,0,8,#5D,#77,#22,0,0,0,0,#88,#DC,#76,#22/A:I;N.
 90 CR.BOT,0,0,0,0,#1F,#1F,#3F,#3F,0,0,0,0,#F8,#F8,#FC,#FC
100 CR.BALK,0,0,0,#FF,#44,#AA,#11,#FF,0,0,0,#FF,#44,#AA,#11,#FF
110 CR.TRAP,0,0,0,#F8,#40,#A8,#10,#F8,0,0,0,#1F,4,#A,#11,#1F
120 CR.LAD,4,4,4,7,4,4,4,7,#20,#20,#20,#E0,#20,#20,#20,#E0
130 CR.SPA,#23,#23,#25,#29,#31,#31,#29,#25,0,0,0,0,0,0,0,0
140 CR.PEP,0,0,#E,#E,#E,4,#E,4,0,0,0,0,0,0,0,0
150 CR.WOLK,0,0,0,1,0,1,0,1,#40,#A0,0,#40,#20,#50,#A0,0
160 CR.UIB1,#F,7,7,1,1,9,6,0,#E0,#B8,#A0,#C0,#C0,#40,#B0,#80
170 CR.UIO1,0,0,0,1,#3,7,7,#F,0,0,0,#C0,#F0,#C0,#E0,#F0
180 CR.MAB,3,1,3,2,2,7,7,#C,#C0,0,#80,#E0,#80,#80,#80,#C0
190 CR.MAO,8,8,#E,3,6,0,6,6,#70,#40,#40,#C0,#20,0,#E0,#E0
200 CR.MEO,3,2,2,3,7,0,7,7,#80,0,#70,#D0,#E0,0,#60,#60
210 CR.HOOP,#F,3,1,#20,0,4,0,0,#F0,#E0,#80,2,0,8,#20,0
220 CR.LEVEL,#D9,#92,#9A,#92,#9A,0,0,0,#36,#A4,#B4,#A4,#B4,0,0,0
230 CR.VORK,#F0,#10,#FF,#10,#F0,0,0,0,0,0,0,0,0,0,0,0
240 P=0;M=3;Z=0;E=0;T=0;K=4;W=4;F=0
250 CLEAR3
260 F.I=0TO112S.16;SET:BALK,I,128;N.
270 F.I=8TO120S.32;SET:BALK,I,32;N.
280 F.I=3TO123S.8;IMAGE:SPA,0,I;N.
290 TURN:SPA
300 F.I=3TO123S.8;SET:SPA,112,I;N.
310 F.I=64TO96S.32
320 F.J=8TO104S.32;SET:TRAP,J,I;N.
330 N.
340 F.I=8TO104S.32
350 F.J=40TO128S.8;SET:LAD,I,J;N.
360 N.
370 F.X=8TO104S.32
380 SET:BALK,X,128;SET:TRAP,X,128
390 N.
400 F.I=0TO#1FFS.4;I!#8000=I!#4200;N.
410 F.I=101TO117S.8
420 SET:MAB,I,167
```

```
430 N.
440 IMAGE:PEP,0,140;TURN:PEP;SET:PEP,112,140
450 SET:LEVEL,0,168
460  GOS.1
470 IF T>3;R=-1
480 F.I=0TO2
490 SET I,24,(96-I*32)
500 SET (I+3),56,(96-I*32)
510 SET (I+6),88,(96-I*32)
520 IF T=0 SET:BOT,(24+I*32),3
530 N.
540 L=1;X=8;Y=144
550 SET:MAB,8,144;SET:MEO,8,136
560 IF T<6 SET:UIB1,8,48;SET:UIO1,8,40
570 DO
580 GOS.a;UNSET:MEO;GOS.n;SET:MAO,X,(Y-8);GOS.p;SOUND100,5
590 GOS.a;UNSET:MAO;GOS.n;SET:MEO,X,(Y-8);GOS.p;SOUND140,5;GOS.b
600 POS 0,I,B;POS 3,I,C;POS 6,I,D
610 U.B=C AND C=D AND D=16
620 F.I=100TO150
630  SOUNDI,30;SOUND(200-I),30
640 N.
650 F.I=0TO8;UNSET I;N.
660 UNSET:UIB1;UNSET:UIO1;UNSET:WOLK
670 UNSET:MAB;UNSET:MAO;UNSET:MEO;UNSET:VORK
680 IF A=255;TU.:MAB;TU.:MAO;TU.:MEO
690 T=T+1;IFT<12 G.460
700 IFM>0 T=0;G.460
710 P.$12"UW SCORE WAS: "Z';E.
720aA=0;B=0;C=0
730qJOYSTK A,B,C;IF A+B=256 OR A+B=2 OR A+B=510;A=0;B=0
740 IF (Y-16)%32<>0 A=0
750 IF (X-8)%32<>0 B=0
760 IF L<>A IFA<>0;TU.:MAO;TU.:MAB;TU.:MEO
770 IFC=1 IFE=0 IFP=1;GOS.k
790 IFA=0 IFB=0 IFT<4 GOS.b;PAUSE12;G.q
800 IFA=0 IFB=0 IFT>3 GOS.b;G.q
810 POS:MAB,X,Y
820 IF X=24 OR X=56 OR X=88 AND Y<140 GOS.h
830 IF (X-8)%32<>0 B=0
840 IF (Y-16)%32<>0 A=0
850 IFX=4 IFA=255;A=0;IF Y=144;PIXEL 5,138,I
    ;IFI;F.I=4TO6;MOVEI,130;PLOT7,I,146;N.;P=1
860 IFX=108 IFA=1 A=0;IFY=144;PIXEL 122,138,I
    ;IFI;F.I=121TO124;MOVEI,130;PLOT7,I,146;N.;P=1
870 IFY=48 IFB=255 B=0
880 IFY=144 IFB=1 B=0
890 R.
900bO=O+1;IF T>5 G.m
910 POS:UIB1,R,S
920 IF T/2=2 GOS.f;G.960
925 I=(R-8)%32;J=(S-16)%32
930 IFI=0 IFJ=0 GOS.f
940 IFI=0 GOS.c
950 IFJ=0 GOS.c
960 POS:WOLK,I,J
970 IF A.(I-R)<4 IF A.(J-S)=4 E=E-1;IF E>0;G.1010
980 IF E=0;UNSET:WOLK
990 IF E>0;E=E-1;TU.:WOLK
```

```
1000 SHO.:UIB1,W,V;SHO.:UIO1,W,V
1010 IF A.(X-R)<8 IFA.(Y-S)<16;GOS.g
1020 SOUND255,7
1030 R.
1040cIFI<>0 V=0
1050 IFJ<>0 W=0
1060 IFR=8 IFW=-4 W=4;TU.:UIB1;TU.:UIO1
1070 IFR=104 IFW=4 W=-4;TU.:UIB1;TU.:UIO1
1080 IFS=48 IFV=-4 V=4
1090 IFS=144 IFV=4 V=-4
1100 R.
1110fIFK<>W IFW<>0 K=W
1120 IFT/2=0 U=A.R.%4;G.s
1130 IFT/2=2 G.r
1140 IFT/2=0 U=A.R.%4;G.s
1150 IFX>R V=0;W=4
1160 IFY<S V=-4;W=0
1170 IFY>S V=4;W=0
1180 IFX<R V=0;W=-4
1190 G.t
1200rIFX>R IFY>S V=4;W=4
1210 IFX>R IFY<S V=-4;W=4
1220 IFX<R IFY>S V=4;W=-4
1230 IFX<R IFY<S V=-4;W=-4
1240 G.t
1250sIFU=0 V=0;W=4
1260 IFU=1 V=4;W=0
1270 IFU=2 V=-4;W=0
1280 IFU=3 V=0;W=-4
1290tIFK<>W IFW<>0;TU.:UIB1;TU.:UIO1
1300 R.
1310gM=M-1;F.I=105TO(105+(3-M)*8-1)
1320  MOVEI,160;DRAWI,167
1330 N.
1340 IF M=0;F.I=100TO150;SOUNDI,30;SOUND(200-I),30;N.;G.710
1350 F.I=0TO10;PAUSEI;?#B002=?#B002:4;N.;PAUSE30
1360 POS:MAO,X,Y;UNSET:MAB;UNSET:MAO;UNSET:MEO
1370 SET:MAB,8,144;IF X=255;SET:MEO,8,136;G.1390
1380 SET:MAO,8,136
1390 X=8;Y=144;UNSET:VORK
1400 R.
1410hI=(X-24)/32*3
1420 POS I,D,J
1430 IF J>Y G.1490
1440 PIXEL(X+5),(Y-18),Q;IF Q=0;G.1490
1450 IF Y/112>0;F.J=1TO16;SHOVE I,0,-2;GOS.i;N.;I=I+1
1460 IF Y/80>0;F.J=1TO16;SHOVE I,0,-2;GOS.i;N.;I=I+1
1470 IF Y/48>0;F.J=1TO(24-(112-Y)/32*4)/2
     ;SHOVE I,0,-2;GOS.i;N.;Z=Z+10;IFY>79;SHO.I,0,-1
1480 F=1
1490 R.
1500iATHIT:UIB1,I;POS:UIB1,R,S;GOS.o
1510 SOUND J,6
1520R.
1530jIF F=1 F=0;R.
1540 POS:MAO,J,I
1550 IF J=255;F.I=16 TO Y S.4;SHO.:MAB,0,-4;SHO.:MEO,0,-4;SOUNDI,5;N.;G.1570
1560 F.I=16 TO Y S.4;SHO.:MAB,0,-4;SHO.:MAO,0,-4;SOUNDI,5;N.
1570 UNSET:MAB;SET:HOOP,X,8;F.N=0TO10;TU.:HOOP;N.;UNSET:HOOP
```

```
1580 G.g
1590kP=P-1;F.I=200TO250S.5;SOUNDI,5;N.
1600 IF L=1 G=X+5;G.1620
1610 G=X-8
1620 SET:WOLK,G,(Y-4);E=30;R.
1630lF.I=17TO20;MOVEI,160;DRAWI,167;N.
1640 IF T/2=0;MOVE19,165;PLOT7,19,161
1650 IF T/2=1;MOVE17,165;PLOT7,19,165;PLOT7,19,163
     ;PLOT7,17,163;PLOT7,17,161;PLOT7,19,161
1660 IF T/2=2;MOVE17,165;PLOT7,19,165;PLOT7,19,161
     ;PLOT7,17,161;MOVE19,163;PLOT7,17,163
1670 IF T/2=3;MOVE17,165;PLOT7,17,163;PLOT7,19,163;MOVE19,164;PLOT7,19,161
1680 IF T/2=4;MOVE19,165;PLOT7,17,165;PLOT7,17,163
     ;PLOT7,19,163;PLOT7,19,161;PLOT7,17,161
1690 IF T/2=5;MOVE17,165;PLOT7,17,161;PLOT7,19,161;PLOT7,19,163;PLOT7,17,163
1700R.
1710mIF R>0 IF T<10 G.1840
1720 IF T/2=5 ;IF R=-1 SET:VORK,128,Y
1730 IF T/2=5 IF Y<S;SHO.:VORK,-8,-4;G.1840
1740 IF T/2=5 IF Y>S;SHO.:VORK,-8,4;G.1840
1750 IF T/2=5 IF Y=S;SHO.:VORK,-8,0;G.1840
1760 IF T/2=4 S=Y-4;R=128;G.1830
1770 IF T/2=3 I=A.R.%3;IF I<>0 G.1920
1780 U=A.R.%4;R=128;S=0
1790 IF U=0 S=43
1800 IF U=1 S=75
1810 IF U=2 S=107
1820 IF U=3 S=139
1830 SET:VORK,R,S
1840 POS:VORK,R,S
1850 IF A=0 IF B=0 PAUSE9
1860 IF R>0 IF T<10 SHO.:VORK,-8,0
1870 IF R=0 UNSET:VORK;R=-1
1880 SOUND 20,4;SOUND 80,4
1890 IF T/2=5 AND R-X=8 OR R-X=4 AND Y-S<10 AND Y-S>-5;R=-1;G.g
1900 IF T/2=5 R.
1910 IF R-X=12 OR R-X=16 AND Y-S<10 AND Y-S>-5;R=-1;G.g
1920R.
1930nIF L<>A IF A<>0 L=A
1940 SHOVE:MAB,(A*4),(B*4)
1950 POS:MAB,X,Y;R.
1960pPIXEL (X+6),(Y-16),Q;IF Q=0;PIXEL(X+5),(Y-18),Q
     ;IF Q=0;PIXEL (X+11),(Y-16),Q;IF Q=0;GOS.j
1970R.
1980oIF J<>10 IF T<4 R.
1990IF R>15 AND R<33 OR R>47 AND R<65 OR R>79 AND R<97 G.2010
2000R.
2010 UNSET:UIB1;UNSET:UIO1;SET:HOOP,(R-4),(S-7)
2020 F.N=0TO40;SOUND(100-N),8;TU.:HOOP;N.
2030 UNSET:HOOP;SET:UIB1,8,48;SET:UIO1,8,40;Z=Z+50
2040R.
```

DE SIMPLEXMETHODE

Operations-research is een wetenschapsgebied dat zich bezig houdt met het vinden van oplossingen voor problemen als "hoeveel verkoopsters moeten er in een winkel staan zonder dat enerzijds de klanten eindeloos moeten wachten voordat ze geholpen worden en zonder dat anderzijds de winkelier een overschot aan personeel hoeft aan te nemen dat alleen in de drukste momenten wat te doen heeft". Dit probleem staat bekend als een "wachttijdenprobleem". Sommige banken hebben nog steeds moeite met de oplossing ervan.
Een ander voorbeeld is dat van de handelsreiziger die een bepaalde route moet afleggen en graag wil weten wat de kortste weg is langs een aantal adressen, waar hij een afspraak heeft.
Ook PERT (zie Acorn nieuws van 1983, nummer 5, blz.49) valt onder operations-research, net als oplossingen voor "job-shop" en "flow-shop" problemen. Dit zijn problemen als "ik heb drie verschillende machines, waarop ik vijf verschillende produkten moet maken. Niet alle vijf de produkten maken in de zelfde volgorde gebruik van deze machines en bovendien legt het ene produkt niet even lang beslag op een machine als het andere produkt. Hoe verdeel ik nu de produkten zo over de machines, dat alles zo snel mogelijk klaar is". Het samenstellen van een schoolrooster lijkt hier een beetje op. Ik hoop hier een volgende keer op terug te komen.
Al dit soort voorbeelden zijn optimaliseringsproblemen. Wachttijden, reistijden, doorlooptijden voor een project, productietijden moeten geoptimaliseerd worden.
De SIMPLEXMETHODE is ook voor zo'n probleem:
Een bierbrouwer fabriceert twee soorten bier(X1 en X2). Op soort X1 maakt de brouwer $13 per vat winst en op soort X2 $23 per vat. Het doel van de brouwer is zijn winst te maximaliseren en de doelstellingsfunctie is daarom

```
13*X1+23*X2
```

Nu zou de brouwer natuurlijk alleen soort X2 kunnen maken. Het probleem is echter, dat er een gebrek aan grondstoffen is. De brouwer heeft slechts de beschikking over 480 pound graan, 160 ounceshop en 1190 pound mout. Voor een vat X1 zijn nodig 5 pound graan, 4 ounce hop en 35 pound mout.Voor X2 is benodigd 15 pound graan, 4 ounce hop en 20 pound mout. Dit betekent dat er drie beperkingen(nevenfuncties) zijn

```
 5*X1+15*X2<= 480
 4*X1+ 4*X2<= 160
35*X1+20*X2<=1190
```

De hamvraag is nu: hoeveel vaten X1 en hoeveel vaten X2 moet de brouwer maken om zijn winst te optimaliseren. Het antwoord wordt door bijgevoegd programma geleverd en is 12 vaten X1 en 28 vaten X2. De winst wordt dan $800.
Het programma vraagt eerst naar de coefficienten van de doelfunctie(13 voor X1 en 23 voor X2). Daarna komt de eerste nevenfunctie aan de beurt: eerst de coefficienten voor X1 en X2 (respectievelijk 5 en 15), daarna de constante(480). Vervolgens hetzelfde voor de tweede en volgende nevenfuncties. Het programma bevat nog een klein schoonheidsfoutje(je ziet het van zelf als je bovenstaand voorbeeld probeert). Wie weet de oplossing hiervoor?
De arrays zijn gedimensioneerd voor maximaal 9 verschillende produkten(soorten bier) en 9 verschillende nevenfuncties.

```
10 PROGRAM SIMPLEX
30 FDIM %ZZ(21),%AA(10,21);DIM XX(10)
40 P.$12"***simplexmethode***"'"OP NUL ZETTEN VAN MATRICES"'
50 F.I=0TO10;XX(I)=0;F.J=0TO 21;%ZZ(J)=0;%AA(I,J)=0;N.;N.
120 N=0;@=0;A=0
140 REM*INVOER VAN GEGEVENS*
150 REM*DOELFUNCTIE*
160 P.$12"***SIMPLEX DOELFUNCTIE***"''
190 P."GEEF IN TE VOEREN COEFFICIENTEN STOP=-1"'
200 F.J=1TO10
210 P."  X"J;FINP.%ZZ(J)
230 P.';FIF %ZZ(J)>=0 G.300
240 FIF %ZZ(J)=-1 G.270
250 P."FOUT COEFFICIENT MOET >=0 ZIJN"';G.210
270 N=J-1;%ZZ(J)=0;J=10
300 N.
310 REM*INVOER NEVENFUNCTIES*
320 F.I=1TO10;P.$12"***SIMPLEX NEVENFUNCTIE "I"***"''
360 P."FUNCTIE "I'
370 P."GEEF IN TE VOEREN COEFFICIENTEN STOP=-1"'
380 F.J=1TON
390 P."  X"J;FINP.%AA(I,J)
410 P.';FIF %AA(I,J)>=0 G.480
420 FIF %AA(I,1)=-1 G.450
430 P."FOUT IN TE VOEREN COEFFICIENT MOET >=0 ZIJN"';G.390
450 A=I-1;J=N;I=10
480 N.
490 IF A<>0 G.550
500 P."GEEF CONSTANTE "I;FINP.%AA(I,21)
520 P.';FIF %AA(I,21)>=0 G.550
530 P."*FOUT* WAARDE MOET>=0 ZIJN"';G.500
550 N.
560 REM*BEREKENING*
570 REM *AANPASSEN DOELFUNCTIE*
580 F.J=1TON;%ZZ(J)=%ZZ(J)*-1;N.
610 REM *TOEVOEGEN SPELINGSVARIABELEN*
620 F.I=1TOA;%AA(I,N+I)=1;XX(I)=N+I;N.
660 REM *GROOTSTE TOEVOEGING-VOLGENDE HOEKPUNT*
670aQ=0;%K=0
690 F.J=1 TO N+A
700 FIF %ZZ(J)>=%K G.730
710 %K=%ZZ(J);Q=J
730 N.
740 REM *Q=0 GEEN GROTERE TOEVOEGING*
750 IF Q=0 G.1190
760 REM *WELKE VARIABELE IS BEPALEND*
770 %V=%AA(1,21)/%AA(1,Q);B=1;F.I=2TOA
800 FIF %AA(I,Q)=0 G.850
810 %C=%AA(I,21)/%AA(I,Q)
820 FIF %C>=%V G.850
830 %V=%C;B=I
850 N.
860 REM*BASIS VARIABELE WISSELEN*
870 XX(B)=Q
880 REM*DOELFUNCTIE BIJWERKEN*
890 %D=ABS(%ZZ(Q)/%AA(B,Q));%E=%AA(B,Q)
910 F.J=1 TO N+A
920 %ZZ(J)=%ZZ(J)+%AA(B,J)*%D;%AA(B,J)=%AA(B,J)/%E
```

```
940 FIF ABS(%AA(B,J))>1E-3 G.960
950 %AA(B,J)=0
960 N.
970 %ZZ(21)=%ZZ(21)+%AA(B,21)*%D
980 FIF ABS(%ZZ(21))>1E-3 G.1000
990 %ZZ(21)=0
1000 %AA(B,21)=%AA(B,21)/%E
1010 FIF ABS(%AA(B,21))>1E-3 G.1040
1020 %AA(B,21)=0
1030 REM*KOLOM Q SCHOONVEGEN*
1040 F.I=1TOA;%F=%AA(I,Q)
1060 IF I=B G.1160
1070 FIF %AA(I,Q)=0 G.1160
1080 F.J=1 TO N+A
1090 %AA(I,J)=%AA(I,J)-%AA(B,J)*%F
1100 FIF ABS(%AA(I,J))>=1E-3 G.1120
1110 %AA(I,J)=0
1120 N.
1130 %AA(I,21)=%AA(I,21)-%AA(B,21)*%F
1140 FIF ABS(%AA(I,21))>1E-3 G.1160
1150 %AA(I,21)=0
1160 N.
1170 G.a
1180 REM*UITVOER*
1190 P.$12"*SIMPLEX-METHODE MAXIMALISATIE*"''
1220 FP."HET OPTIMUM LEVERT "%ZZ(21)''
1240 F.I=1TOA;P."X"XX(I)"=";FP.%AA(I,21)';N.;E.
```

Floppy-Disk-Controller

Verplaatsen in memory-map.

De Floppy-Disk-Controller zit normaal op de nogal ongelukkige plaats #0Axx. Als we 'm daar weg willen hebben, dan moet de DOS-EPROM en de hardware worden veranderd.
De DOS wordt veranderd door op alle plaatsen waar de FDC wordt aangeroepen het adres te veranderen in z'n nieuwe adres. Hieronder staan die adressen van de DOS.
E004, E009, E228, E22D, E7DD, E7E2, E7E6, E7EB, E80C, E814, E851, E85F, E87D.
Op die adressen staat nu dus #0A. Als we dat veranderen in bv. #BE, dan wordt vanaf dan de FDC daar gezocht.
De hardware aanpassing kan dan gemaakt worden, na de DOS aanpassing dus. Het kan nl. nodig zijn de Flop te gebruiken om de nieuwe EPROM te maken. Nog een advies: Wis de oude DOS pas als je zeker weet dat de Flop altijd goed werkt, met alle programma's. Stel je voor het een of andere routinetje rommelt wat op #BExx, dan gaat er vast iets fout.
Die aanpassing is simpel. Verbreek van PL6/7 punt 31 (inv. block 0) en verbind dat punt met ic23 pin 12 (#Bxxx). Op de FDC-print wordt nu nog ic22 (74LS138) pin 14 losgesneden en het spoortje verbonden met pin 12. Nu staat de FDControler op #BExx.
Dat is alles.

WAARSCHUWING.

Ik heb de FDC niet op die plaats geprobeerd. Ik draai op #BFF4- #BFFB, dat kwam bij mij het beste uit.

INITIALISATIE ROUTINE.

Adressen voor de aanpassing van de DOS voor andere drive's:

adres	nu	funktie	TANDON	TEAC
E864	#14	#14*2msec. steprate.	#0A	#03
E865	#05	5*2 msec. headsettlingtime	#0A	#0A
E866	#CA	#C omwentelingen na aktie voordat de motor stopt #A*8 msec headload setlingtime	#7A	#CA

Om voor het aanpassen van de eprom de nieuwe waarden te proberen is het volgende programma.

```
10 *DOS     : initialiseer DOS
20 A=#A00   : adres van de controller.
30 ?A=#35   : specify commando
40 A?1=#D   : initialiseer parameter.
50 A?2=#0A  : step-rate.
60 A?3=#0A  : headsetlingtime.
70 A?4=#7A  : omwentelingen / head loadtime.
80 END
```

Na een break of *DOS moet dit programma opnieuw worden gerund.

```
10 REM INTELLIGENTE DISAS
20 REM F.LE BLANC
30 REM BASISDISAS IS VAN F.V.HOESEL
40 DIM LL20,TT2,PP7,VV16
50 IN."ROUTINE OP: "Q
60 FORI=0TO50;LLI=0;N.
70 P.$21;@=0
80 P=Q;GOS.a;P=Q;GOS.a
90 P.$6$11"ROUTINE OP: #"
100 P.&Q"-#"&P'
110 END
120    zeropagegebruik
130 90,91  : DISAS-ADRES
140 92     : AANTAL INSTR.
150 93,94  : ASCIIPRINT
160 95     : TESTBYTE PRINTER
170 96     : STACKPOINTER JSR
180 140-1BF: JSR-STACK
190a[
200        JSR#C4E1;LDY@#90
210        LDX#4;JSR#C3CD
220        JSR PP6\test printer
230        LDA@#80;STA#96
240:LL0
250        JSR LL1;JSR VV1
260        JMP LL0
270:LL1
280        LDA#90;STA#93
290        LDA#91;STA#94
300        LDY@2
310:LL2
320        LDA(#90),Y;STA#66,Y
330        DEY;BPL LL2
340:LL3
350        INY;STY#F;LDX@#40
360:LL4
370        LDA#66;CMP@#D4
380        BEQ LL6;SEC
390        SBC#F1F3,Y;SEC
400        SBC#F210,X;BNE LL6
410        STA#65;LDY#F250,X
420:LL5
430        ROR A;ROR#65;DEY
440        BNE LL5;LDY#F
450        AND#F1D5,Y
460        BNE LL7;LDA#F1E4,Y
470        AND#65;BNE LL7
480:LL6
490        DEX;BNE LL4;CPY@#E
500        BNE LL3;LDY@0
510        LDX@#11;STY#F
520:LL7
530        LDA#F202,Y;CMP@#F
540        BNE LL8;LDA@1
550:LL8
560        AND@#F;STA#69
570        STA#92\aant.instr.
580        STA#0;STX#6A
590        LDX@#90;JSR#F7F1
600        \F7F1: p.adres
610        LDX@#90;LDY@0
620:LL9
630        LDA (#90),Y;JSR#F7FA
640        \F7FA: p.adresinhoud
650        JSR#FA08;BNE LL10
660        PLA;PLA
670:LL10
680        DEC#69;BPL LL9
690        LDY#E0;JSR#F99A
700        \F99A: p.spaties
710        JSR PP0\p.ascii
720        LDX#6A;LDA#F194,X
730        STA#64;LDA#F154,X
740        STA#65;LDY@3
750:LL11
760        LDX@5
770:LL12
780        ROL#64;ROL#65;ROL A
790        DEX;BNE LL12
800        AND@#1F;CLC
810        ADC@#3F
820        CMP@#3F;BNE LL13
830        LDA@#2D
840:LL13
850        JSR#FFF4
860        DEY;BNE LL11
870        JSR#F7FD;LDY@8
880        LDX#F;LDA TT0,X
890        STA#69
900:LL14
910        CPY@6;BNE LL19
920        TXA;BNE LL16
930        LDA#67;BPL LL15;DEX
940:LL15
950        CLC;ADC#90;STA#67
960        TXA;ADC#91;STA#68
970        LDX@2;BNE LL17
980:LL16
990        LDX0;BEQ LL19
1000:LL17
1010       LDA@#23;JSR#FFF4
1020:LL18
1030       LDA#66,X;JSR#F802
1040       DEX;BNE LL18
1050:LL19
1060       ASL#69;BCS LL20
1070       LDA TT1,Y;JSR#FFE9
1080:LL20
1090       DEX;DEY;BPL LL14
1100       LDA@#80;STA#E1
1110       JMP#C504
1120:PP0   \ascii-dump
1130       LDY@#FF
1140:PP1
1150       INY;LDA(#93),Y
1160       CMP@#20;BCC PP2
1170       CMP@#7F;BCC PP3
1180:PP2
```

```
1190      LDA@#2E
1200:PP3
1210      JSR#FFF4
1220      DEC#92;BPL PP1
1230:PP4
1240      JSR#F7FD;INY
1250      CPY@4;BNE PP4
1260:PP5
1270      RTS
1280:PP6  \test printer
1290      LDA#B80C;BEQ PP7\uit
1300      LDA@1            \aan
1310:PP7
1320      STA#95;RTS
1330:VV1  \branch of jump
1340      LDX@24;LDY@0
1350      LDA(#93),Y
1360:VV2
1370      DEX;DEX;BMI VV5
1380      CMP TT2,X;BNE VV2
1390      INX;LDA TT2,X
1400      BEQ VV3        \jsr
1410      CMP@1;BEQ VV6\rts
1420      CMP@2;BEQ VV7\branch
1430      CMP@3;BEQ VV10\jmp
1440      CMP@4;BEQ VV11\jmp()
1450:VV3
1460      JSR VV12;BCS VV5
1470      LDX#96;BEQ VV5
1480      DEC#96;DEC#96;LDX#96
1490      LDA#90;STA#140,X
1500      LDA#91;STA#141,X
1510:VV4
1520      LDY@1
1530      LDA(#93),Y;STA#90
1540      INY;LDA(#93),Y
1550      STA#91
1560:VV5
1570      RTS
1580:VV6
1590      LDX#96;BMI VV5
1600      JSR#FFED
1610      INC#96;INC#96
1620      LDA#140,X;STA#90
1630      LDA#141,X;STA#91
1640      RTS
1650:VV7
1660      JSR VV12;BCS VV5
1670      LDY@1;LDA(#93),Y
1680      BMI VV9
1690      CLC;ADC#90;STA#90
1700      BCC VV8
1710      INC#91
1720:VV8
1730      RTS
1740:VV9
1750      STA#92;LDA@0
1760      SEC;SBC#92;STA#92
1770      LDA#90
1780      SEC;SBC#92;STA#90
1790      BCS VV8
1800      DEC#91;RTS
1810:VV10
1820      JSR VV12;BCS VV5
1830      JMP VV4
1840:VV11
1850      JSR VV12;BCS VV5
1860      LDY@1
1870      LDA(#93),Y;STA#92
1880      INY;LDA(#93),Y;STA#93
1890      LDY@0
1900      LDA(#92),Y;STA#90
1910      INY;LDA(#92),Y;STA#91
1920      RTS
1930:VV12
1940      LDA@3;JSR#FEFB
1950:VV13
1960      LDA@0;STA#E1
1970      JSR#F7D1
1980];$P="DO YOU WANT TO JUMP [Y/N] "
1990P=P+L.P;[
2000      NOP;JSR#FFE3;PHA
2010      JSR#FE22;LDA@#B
2020      JSR#FFF4;JSR#FFED
2030      LDA#95;BEQ VV14
2040      LDA@2;JSR#FEFB
2050:VV14
2060      PLA
2070      CMP@CH"Y";BNE VV15
2080      JSR#FFED
2090      CLC;RTS
2100:VV15
2110      CMP@27;BNE VV16
2120      LDA@#80;STA#E1
2130      JMP#C55B
2140:VV16
2150      SEC;RTS
2160]
2170 TT(0)=P
2180 !P=#F9FFFEFF;P!4=#7F7FFFCF
2190 P!8=#87B1CFF9;P!12=#F9B7FF
2200 P=P+15;TT(1)=P
2210 !P=#0D;P!1=#292C5941
2220 P!5=#40282C58;P=P+9
2230 TT(2)=P
2240 !P=#34C046C  ;REM jmp
2250 P!4=#2500270 ;REM bvc,bvs
2260 P!8=#2300210 ;REM bmi,bpl
2270 P!12=#29002B0;REM bcc,bcs
2280 P!16=#2D002F0;REM bne,beq
2290 P!20=#0200160;REM jsr,rts
2300 P=P+24
2310 RETURN
```

Deze disas print ook de ascci-waarden.

Bij branche's en jumps vraagt de disas of er gesprongen moet worden. Geantwoordt moet worden met de toets "Y" voor een sprong of een branch. Bij iedere andere toets wordt de disas vervolgd.

Maximaal zijn er 64 JSR-sprongen.

In Acorntjesbrood 2.3 heb ik op pag.18 beschreven dat ik de Wordpack heb omgebouwd naar een beeldscherm van 64 kolommen en 24 regels m.b.v. de VDU-64 van Jos Horsmeier. De toen gepresenteerde versie (release 0.0) kan niet in een 4K eprom worden gezet, omdat er van #9800 tot #9B00 een tabel met de karakterset voor de VDU-64 staat, die je dan dus steeds apart moet inladen, wat voor niet-floppy-disk-bezitters zoals ikzelf nogal lastig is. Deze tabel kan niet in de Wordpack zelf worden opgeslagen wegens plaatsgebrek. Nu is het zo dat deze tabel dubbel is uitgevoerd, en dus ook in anderhalve geheugenpagina zou kunnen worden opgeslagen. Voor zo'n compacte tabel is wel plaats in de Wordpack, maar dan moet ik de VDU-64 routines ingrijpend wijzigen, waarvoor de tijd en zin (vooral dat laatste) ontbreken. Toch wou ik (en anderen ook) de 64 kolommen Wordpack graag in een eprom hebben, als vervanging van de normale Wordpack, die al geruime tijd werkloos op de schakelkaart zat.

In de hier gepresenteerde versie (release 0.2) is voor een m.i. zeer aanvaardbaar compromis gekozen: De compacte tabel van anderhalve pagina met de karakterset is in de 4K Wordpack opgeslagen, die bij binnenkomst van de Wordpack (dus na intypen van ED64) wordt gekopiëerd en tegelijk uitgevouwen tot 3 pagina's naar #9800-#9B00.
Dit gaat zo snel dat je het niet eens merkt en als je RAM hebt gestapeld van #9800-#9FFF (dit gebied wordt bij mij alleen als scratchpad gebruikt) is deze versie te beschouwen als een 4K Wordpack die in eprom kan worden gezet. Hij is dus wel op te slaan binnen 4K, maar draait niet binnen 4K.

Behalve het uitklappen van de tabel met karakters zijn in deze versie (release 0.2) de volgende wijzigingen aangebracht t.o.v. release 0.0 (voor de wijzigingen van release 0.0 t.o.v. de normale Wordpack zie Acorntjesbrood 2.3 pag.18.):

- Gewijzigd 'Q' (Quit)-commando.
  Na het indrukken van Q (Quit) wordt in de normale Wordpack de tekstfile geconverteerd naar een Basic programma. Dit is in de meeste gevallen erg hinderlijk omdat de Wordpack meestal voor tekstverwerking wordt gebruikt. In deze versie is het mogelijk om naar keuze al dan niet de tekstfile te converteren. Dat gaat als volgt:
  Na het geven van het Q-commando wordt naar het 'normale' scherm teruggeschakeld en wacht de Wordpack op een toets aanslag. Als U op 'C' (Convert) drukt, wordt de tekstfile naar Basic geconverteerd, zoals dat normaal ook het geval is. Drukt U op een andere toets, dan wordt direct de Wordpack verlaten en bevindt U zich in de Basic-text-space vanaf #8200, waar een NEW wordt uitgevoerd.
  Gewijzigd 'O' (Output) commando.
  Om de uitvoer die bij het 'O' commando op de printer wordt afgedrukt, ook op het beeldscherm te kunnen bekijken, is in Acorntjesbrood 1.2 op pag.7 een zgn. 'Papierspaarder' beschreven. Deze is in een iets uitgebreidere vorm in deze Wordpack opgenomen.

Hiermee werken we als volgt:
Na het geven van het 'O'-commando wacht de Wordpack op een toetsaanslag. Als U nu op 'P' (Printer) drukt, verschijnt de uitvoer op de printer, net zoals dat normaal het geval is na 'O'. Drukt U op een andere toets, dan verschijnt de uitvoer op het scherm.
Bij elke nieuwe pagina hoort U een piepje en wacht de editor op een toetsaanslag alvorens verder te gaan met het afdrukken van de uitvoer.
Als de hele tekstfile is afgedrukt (of het scherm of op de printer), wordt dat aangegeven met 2 piepjes . Nu wacht de Wordpack op een toetsaanslag zodat U de laatste pagina ook rustig kunt bekijken. Hetzelfde gebeurt als U het afdrukken onderbreekt met de ESC-toets.
De uitvoer is te stoppen met de SHIFT-toets (met de S van stop) en kan weer worden doorgestart met de CTRL-toets (met de C van continueer). Drukt U zowel SHIFT als CTRL in, dan wordt het afdrukken van een karakter sterk vertraagd, zodat U op Uw gemak kunt meelezen.
Met het .j commando is het mogelijk om een rechte rechterkantlijn te maken. Hierbij kan het af en toe gebeuren dat, als er in de tekst een-erg-lang-woord-voorkomt-zoals-dit, de ruimtes tussen de woorden wel èrg groot worden. (Deze zin lijkt me wel een goed voorbeeld.) Doordat het nu mogelijk is om van te voren precies te kunnen zien hoe de tekst op papier verschijnt, kan dit voorkomen worden.

- Nette layout na 'Paper'.
  Bij het printen van de tekst via het 'O'-commando drukt de Wordpack bij elke nieuwe pagina het woord 'Paper' af en wacht op een toetsaanslag, als U tenminste .k heeft gebruikt. Dit om bij gebruik van losse velletjes de gebruiker de kans te geven een nieuw vel in te draaien.
  Helaas wordt hierd [illegible]
  beeldscherm versch [illegible]
  voorkomen worden n [illegible]
  nodig) een toets is aangeslagen, 5 Delete-karakters (ASCII-waarde #7F) afgedrukt. Het woordje 'Paper' zal daardoor weer verdwijnen. Heeft U .k niet gebruikt, dan zal 'Paper' meteen weer verdwijnen, waardoor het lijkt of het niet is afgedrukt.

Release 0.2 is t.o.v. release 0.0 intern behoorlijk gewijzigd. In release 0.0 was nog aardig wat plaats vrij, die in release 0.2 helemaal is gebruikt. De hele eprom is dus vol.
(De nummering release 0.0 en 0.2 doet vermoeden dat er ook een release 0.1 is. Die is er inderdaad even geweest, maar werd al snel opgevolgd door release 0.2. In de welkomstekst na ED64 wordt ondermeer dit versienummer afgedrukt.)
De hier gepresenteerde Wordpack is voor niet-80-koloms-kaart bezitters een (vind ik zelf) behoorlijke verbetering van de mogelijheden van de Atom als tekstverwerker, doordat je, als je een maximale breedte van 63 karakters aanhoudt (zoals deze tekst), precies je tekst kunt bekijken zoals die op papier zal verschijnen. Ook 80-koloms-kaart bezitters kunnen hun voordeel ermee doen, vanwege de ingebouwde faciliteiten om de uitvoer op de printer te bekijken zonder een al te groot papierverbruik.

DECADOS

Decados bestaat uit een tiental uitbreidingen op de normale ATOM-dos, die gerealiseerd zijn op een manier die de compatibiliteit met de ATOM-dos zo groot mogelijk maakt. Die compatibiliteit is verkregen door geen van de bestaande routines te verschuiven; de start-adressen zijn dus niet veranderd.

De eerste zin van dit verhaal is met opzet voorzichtig geformuleerd, omdat er uiteraard wijzigingen nodig waren om decados te kunnen programmeren. Een van die wijzigingen is het verwijderen van de VDU-routines. Het VDU-commando bestaat nog wel en wordt dus nog steeds correct uitgevoerd, maar de read- en write vectoren wijzen nu niet meer naar een bestaande routine (bedoeld voor de originele VDU-kaart van ACORN), maar naar een JMP (#2200) en JMP (#2202). Deze adressen worden niet door de dos gebruikt en kunnen dus eventueel gevuld worden met adressen die wijzen naar een VDU-routine, die dan elders in RAM moet staan. Hier kan natuurlijk de originele ATOM VDU-routine staan (voor de viditel-kaart), maar dat mag ook een eigen geschreven routine zijn voor een andere kaart (80-koloms kaart). Indien u het VDU-commando niet gebruikt hoeft u #2200-#2203 niet te vullen. Samengevat: nadeel: geen originele routine in de dos-rom (is echter nog in ram te plaatsen); voordeel: mogelijkheid voor andere routine (voor andere kaarten) en vooral een hoop lege ruimte in de dos-rom.

De ruimte die zowiezo al leeg was (in de ACORN-versie van de dos) is natuurlijk ook in gebruik genomen door decados. Hierdoor vervalt de mogelijkheid van *OPT (uit de TELEC-versie), maar die is nu ook niet meer zo nodig.

tien voordelen

1. Er zijn een zestal nieuwe dos-commando's gemaakt:

*RENAME <filenaam> <filenaam>
geeft een nieuwe naam aan een bestaande file.
voorbeeld:

*RENAME JAAP JOEP
de filenaam JAAP verandert in JOEP

*INFALL (<drive-nummer)
geeft een lijst van alle files met hun adressen.
Het drive-nummer mag weggelaten worden. Wordt het drivenummer toch gebruikt, dan wordt de drive met dat nummergeselecteerd.

*COS
schakelt over van DOS naar de standaard COS
Eventueel daarna dus nog overschakelen naar 1200 Baud (COS 1).

*DOS
reset het Disk Operating Sysem
Er komt dus geen foutmelding als *DOS tweemaal gegeven wordt.
Als bijverschijnsel kan het ook gebruikt worden om de drive te stoppen.

*TYPE <filenaam>
de file wordt als ASCII-file afgedrukt.
Dit commando is handig voor het bekijken van textfiles en macro-files (zie verder). Het is niet geschikt om BASIC programma's te bekijken, omdat daar controltekens in voorkomen (regelnummers staan niet in ASCII-code, maar als getal in het geheugen).

*CREATE <filenaam>
voor het aanmaken van textfiles en macro-files.
Na create verschijnt er een dubbele prompt '>>' en vanaf dat ogenblik wordt alles wat u intikt in de file opgeborgen.
Een uitzondering hierop is DELETE. De routine wordt gestopt door ESC of <CTRL D> (=EOF, End Of File).
Als de file reeds bestaat volgt de melding EXISTS. De oplossing hiervoor is de file te DELETEN of te RENAMEN.

2. Als tweede uitbreiding is er nu de mogelijkheid om een qualifier te gebruiken in een filenaam. Voorbeeld:

```
*LOAD "B:OPSLAG"
```

De file wordt nu gezocht onder de qualifier 'B'. Dit werkt vele malen gemakkelijker dan *USE. Ook is het nu mogelijk om de qualifier van een file te wijzigen. Voorbeeld:

```
*RENAME B:OPSLAG D:OPSLAG      of bv.
*RENAME B:OPSLAG :OPSLAG
```

in het eerste geval verandert de qualifier van 'B' naar 'D' en in het tweede geval van 'B' naar een spatie.

3. Er is nu ook een zg. INFO-switch. Deze switch kan bij ieder commando worden gegeven, maar bij sommige commando's heeft dat geen effect. Hoe gaat dat? Voorbeeld:

```
*LOAD/I   STRIPS
*DELETE/I PMD
*DEL./I   PMD
```

De schuine streep (voor switch) en de letter I (de naam van de switch) moeten direct achter het commando staan. Het resultaat is dat *MON aan staat voor de duur van dat ene commando.

4. Naast de I-switch is er ook een M-switch. Deze kan echter alleen bij *CAT gegeven worden. Voorbeeld

```
*CAT/M
*./M
```

Indien deze switch gebruikt wordt, gaat de schijf niet draaien, maar wordt een catalog uit het geheugen (Memory) gehaald.

5. Bij het *SAVE commando mag gebruik worden gemaakt van optelling. Voorbeeld:

```
*SAVE QUEUE 3000 +2340 +50
```

Dit komt overeen met:

```
*SAVE QUEUE 3000 5340 3050
```

Het is vooral handig als je wel de lengte en het start-adres van een file weet, maar niet het eindadres.

6. Bij het gebruik van een filenaam als pseudo-commando bepaalt nu het executie-adres wat er gebeurt. Is dit adres ongelijk aan #0000 dan gaat alles zoals het ging in de ATOMdos. Is het echter wel gelijk aan nul, dan wordt is de file blijkbaar geen machinetaal programma (die beginnen nooit op #0000), maar een random-acces- of macro-file. Zo'n file wordt nu niet gerund, maar uitgevoerd mbv *EXEC.
Dat dit zeer handig kan zijn blijkt uit het volgende voorbeeld:

```
>*CREATE DEMO
>>*NOMON
>>*LOAD DATA 8200
>>*LOAD BASE 4000
>>?18=#40
>>OLD
>>RUN
```

```
>>>                 (ESC of <CTRL D>)
>
>*DEMO              (is nu gelijk aan *EXEC DEMO)
```

PASOP: EXEC gebruikt de read-vector (slaat de oude vector wel op) en werkt daardoor niet met CHARON samen.

7. Het *GO commando is verbeterd. In de gedis-assembleerde ATOM-dos is duidelijk te zien dat de programmeur het volgende mogelijk vond:

```
*LOAD RELOC  gevolgd door:
*GO
```

Een *GO zonder adres start het laatst ingeladen programma op het bijbehorende executie-adres. Door een kleine denkfout werkte dat echter niet. Nu dus wel.

8. De catalog verschijnt nu over drie kolommen, zodat de inhoud van een schijf met veel files nu ook op een scherm past.

9. Voor nummer negen was helaas geen ruimte meer over in de EPROM. Die ruimte kunnen we nu echter benutten voor het maken van een opmerking: alle routines in decados maken geen gebruik van de BASIC-, Floatingpoint-, of utility-ROM. Hierdoor kunnen ook systemen met BASIC in RAM gebruik maken van decados (bij mij zelf zit alles nog gewoon in ROM, maar je weet maar nooit. Compatibel met de toekomst).

10. Haakjes, kort maar krachtig.
Enkele voorbeelden (hopelijk duidelijker dan tien regels tekst):

```
*DELETE FILE(1 2 3)                 *DELETE FILE1
                                    *DELETE FILE2
                                    *DELETE FILE3
*(UNLOCK DELETE) PROTECT            *UNLOCK PROTECT
                                    *DELETE PROTECT
*RENAME D(OO AA)R (AAP MIES)JE      *RENAME DOOR AAPJE
                                    *RENAME DAAR MIESJE
```

Haakjes maken het dus mogelijk om iets meerdere keren te doen, waarbij er dan steeds net iets anders moet gebeuren (andere filenaam, ander commando oid.).
De ronde haakjes mogen niet worden genest (resultaten zijn dan wel voorspelbaar, maar waarschijnlijk niet wat u voor ogen had).

Mensen in het bezit van een mini-schakelkaart kunnen een en ander zo uitproberen door hun RAM naar het gebied #Exxx te schakelen, maar uiteindelijk is het de bedoeling om decados in eprom te zetten en de ATOM-dos eprom op de controller-kaart te vervangen door de decados eprom.

```
  10 PROGRAM DECADOS
  20 REM             by FRANS
  30
  40 REM SORRY, NOG UIT DE TIJD DAT IK GEEN MINI-ASSEMBLER HAD.
  50 REM  EEN ROMMELTJE DUS !!!
  60
  70 Q=?#B001&#80;REM SHIFT?
  80
  90 DIM LL40,EE1,MM40,NN40,KK40
 100 FOR I=0 TO 40;LL(I)=999;MM(I)=999;NN(I)=999;KK(I)=999;NEXT
 110 COPY #E000,#EFFF,#5000
 120 B=#E0-#50;C=#E000-#5000
 130 T=#EF64-1
 140 PRINT $21;GOSUB a
 150 IF Q P.$6
 160 GOSUB a;PRINT $6
 170 ?#53E3=#EF;?#53E4=#3A
 180 ?#56FB=#25;REM READVDU
 190 ?#5EFA=NN0%256;?#5EFF=NN0/256+B;REM HAAKJES
 200 ?#5386=KK8/256+B;?#5387=KK8
 210 ?#545A=#FE;REM NOMON
 220 RAM;REM MINISCHAKELKAART RAM OP #E000
 230 COPY #5000,#5FFF,#E000
 240 *DOS
 250 END
 260aP=#5F3A;[
 270:LL0;DEY;STY #9A;
 280:LL1;LDY #9A;
 290:LL2;INX;
 300:LL3;INY;LDA T,X;BMI LL5;CMP #100,Y;BEQ LL2;
 310:LL4;INX;LDA T,X;BPL LL4;INX;LDA #100,Y
 320CMP @#2E;BNE LL1;DEX;BCS LL3;
 330:LL5;STA #9B;LDA T+1,X;JMP NN31+C
 340]
 350 $P="RENAME";P=P+LEN(P);?P=LL6/256+B;P?1=LL6%256;P=P+2
 360 $P="INFALL";P=P+LEN(P);?P=LL8/256+B;P?1=LL8%256;P=P+2
 370 $P="COS";P=P+LEN(P);?P=LL10/256+B;P?1=LL10%256;P=P+2
 380 $P="DOS";P=P+LEN(P);?P=#E0;P?1=#00;P=P+2
 390 $P="TYPE";P=P+LEN(P);?P=LL12/256+B;P?1=LL12%256;P=P+2
 400 $P="CREATE";P=P+LEN(P);?P=LL15/256+B;P?1=LL15%256;P=P+2
 410 ?P=#E4;P?1=#C5;P=P+2
 420[
 430\infall
 440:LL8;JSR #E231;LDY #2105;
 450:LL9;JSR #E109;TYA;PHA;JSR #E1BF;PLA;TAY;BNE LL9;RTS;
 460\memory only
 470:KK12;JSR #FB76;CMP @#2F;BNE KK13;LDA #101,Y;CMP @#4D
 480BNE KK13;INY;INY;JMP #E5CC
 490:KK13;JMP #E231
 500\go
 510:KK6;LDX @#9B;JSR #E0B6;JMP #E568;
 520\save+
 530:KK8;JSR #E041;JSR #E084;BEQ KK9;LDX @#A2;JSR KK10+C
 540BEQ KK9;LDX @#9E;JSR KK10+C;JMP #E5FA;
 550:KK10;JSR #FB76;CMP @#2B;BNE KK11;INY;JSR #E0B6;PHP
 560;CLC;LDA 0,X;ADC #9C;STA 0,X;LDA 1,X;ADC #9D;STA 1,X
 570PLP;RTS
```

```
580:KK11;JMP #E0B6;
590:KK9;JMP #E503;
600];$P="DECADOS.3";P=#556F;[
610\go
620JMP (#009B)
630];?#53D5=(KK6+C)/256;?#53D6=(KK6+C);[
640];P=#5D22;[
650\writevdu
660:EE0;JMP (#2200)
670\readvdu
680:EE1;JMP (#2202);
690\rename
700:LL6;LDA #AC;PHA;JSR #E033;JSR #E14C;PLA;STA #AC
710TYA;PHA;JSR #E18B
720LDX @#40;STX #9A;LDX @#9A;JSR #E068
730PLA;TAY;JSR #E100
740LDA #2007,Y;AND @#80;ORA #AC;STA #2007,Y;LDX @6
750:LL7;LDA #A5,X;STA #2006,Y;DEY;DEX;BPL LL7
760JMP #E5A9
770\cos
780:LL10;JSR #E5CC;LDX @#F
790:LL11;LDA #FFA2,X;STA #20C,X;DEX;BPL LL11
800LDA @#EF;STA #206;LDA @#F8;STA #207;RTS;
810\macro
820:MM0;
830LDA #210A,Y;ORA #210B,Y;BNE MM2;LDY @#00;JSR #E5C9
840JSR #FFCE;TAY;JSR #E500;JMP #E520
850:MM2;JSR #E487;JMP #E4F7
860\type
870:LL12;JSR #E5C9;JSR #FFCE;TAY;BNE LL13;JMP #E516;
880:KK7;JSR #FFE9;
890:LL13;JSR #FFD4;BCC KK7
900:LL14;JSR #FFCB;JMP #E460;
910\create
920:LL15;JSR #E5C9;JSR #FFCE;TAY;BEQ LL26
930JSR #E016;];$P="EXISTS";P=P+LEN(P);[;BRK;
940:LL26;CLC;JSR #FFCE;TAY
950:LL16;JSR #FFED;
960:LL21;LDA @#3E;JSR #FFF4;JSR #FFF4;
970:LL17;LDX @#3F;
980:LL18;INX;BPL LL19;
990:LL24;JSR #FFE3;CMP @#7F;BNE LL24;JSR #FFF4
1000:LL20;DEX;
1010:LL19;JSR #FFE6;CMP @#18;BEQ LL16;CMP @#7F;BEQ LL20
1020CMP @#1B;BEQ LL14;CMP @#04;BEQ LL14
1030STA #100,X;CMP @#0D;BNE LL18
1040LDX @#40;
1050:LL25;LDA #100,X;JSR #FFD1;INX;CMP @#0D;BNE LL25;BEQ LL21;
1060\angle brackets
1070:NN20;DEX;
1080:NN1;JSR NN6+C;AND @#FF;BMI NN1
1090JSR NN2+C;BEQ NN20;
1100:NN14;CMP @#22:#80;BNE NN11;
1110:NN9;INY;LDA #22C0,Y;
1120:NN10;JSR NN13+C;JSR NN2+C;CMP @#22:#80;BNE NN9;
1130:NN11;JSR NN6+C;JSR NN2+C;BNE NN14;DEX;
1140:NN15;JSR NN6+C;BNE NN15;
```

```
1150:NN6;INY;LDA #22C0,Y;
1160:NN12;CMP @#29;BEQ NN5;
1170:NN13;CMP @#0D;BNE NN4;
1180JMP #E5D3;
1190:NN5;PLA;PLA;
1200:NN4;RTS;
1210:NN2;STA #100,X;INX;
1220:NN8;ORA @#80;STA #22C0,Y;CMP @#A0;BEQ NN3;STA #9A;
1230:NN3;RTS;
1240\info switch
1250:NN38;LDA #C0;AND @#01;BNE NN39
1260LDA #EF;BNE NN3
1270:NN39;JMP #E1BF
1280\
1290:NN0;LDX @#40;STX #9A
1300:NN18;LDA #1FF,X;];P?-1=0;[;STA #22BF,X;DEX;BNE NN18;
1310:NN19;LDY @#FF;
1320:NN16;INY;LDA #22C0,Y;CMP @#28;BNE NN17
1330JSR NN1+C;JMP NN16+C;
1340:NN17;STA #100,X;INX;CMP @#0D;BNE NN16
1350LDA #9A;BEQ NN4
1360:NN23;JSR #E3E5;LDX @0;STX #9A;BEQ NN19;
1370\qualifier
1380:KK0;CMP @#3A;BNE KK2;CPY @1;BCC KK5;BNE KK2;
1390:KK5;LDA #A5;STA #AC;LDA @#20;STA #A5
1400:KK4;INC #9A;BNE KK1;INC #9B;
1410:KK1;DEY;BPL KK4;BNE KK3;
1420:KK2;STA #00A5,Y;
1430:KK3;JMP #E08D;
1440\switch info
1450:NN35;LDA #E36D,X;
1460:NN31;PHA
1470:NN32;LDA #100,Y;CMP @#2F;CLC;BNE NN34;LDA #101,Y;CMP @#49
1480CLC;BNE NN34;INY;INY;SEC;
1490:NN34;PHP;LDA #C0;LSR A;PLP;ROL A;STA #C0;PLA;JMP #E412
1500];P=#508A;[
1510\qual
1520JMP KK0+C;
1530];P=#54F4;[
1540\macro
1550JMP MM0+C
1560];P=#5237;[
1570\memory
1580JSR KK12+C
1590];P=#540F;[
1600\switch
1610JMP NN35+C
1620];P=#51BB;[
1630\info switch
1640JMP NN38+C
1650];P=#5293;[
1660JSR #E0EC;DEC #B8;BPL LL27;JSR #FFED;JSR #E0EC;BNE #530D;
1670:LL27;
1680];P=#5310;[
1690LDY @2;STY #B8;JSR #E0F3;
1700]
1710 RETURN
```

Acorn-Atom meets Aquarius.

Waarschijnlijk doordat de Aquarius-computer de markt gaat verlaten, zijn er van dat merk een aantal uitbreidingen te koop voor prijzen die te gek zijn.
Een voorbeeld, 16k geheugen-module (RAM dus) voor f 89,-.
Wat heeft dat nu voor de Atom te betekenen? Wel de Aquarius heeft een 1Mhz 6502, net als de Atom. Het zou dan ook vreemd zijn als RAM van de een niet op de ander "draait". Een gokje blijft het natuurlijk wel; er kunnen stroomvreters gebruikt zijn, of dynamische RAMs. Dat gokje heb ik genomen (toen 16k nog f.139,- kostte) en met geluk.
In die modules zit de HM6116 statische CMOS RAM. In beginsel geschikt voor battery-backup.
Om bij het begin te beginnen.
Voor een van de genoemde bedragen koop je bij FUNTRONICS (nee meneer Keizer, daar heb ik ook geen banden mee) een kartonnen doos van 16x11x5cm. Daar komt dan een houder uit met de module. Die module is van plastic en meet 7x7,5x4,5cm. Twee schroeven geven toegang tot het elektronisch-centrum, dat echter nog wel is verpakt in blik. Wat je overhoudt is een printje van 66x64x8mm.
Hoe krijg je daar nu 8 stuks 6116 op, zal de snelle rekenaar vragen; een 6116 is immers net zo groot als een EPROM(32x15x5mm) !?.
Dat klopt maar in de geautomatiseerde bestukkings-industrie is de FLAT-PACK behuizing aan het terrein-winnen. De 6116 die hier gebruikt is heeft dan ook de toevoeging FP 4 achter 6116 staan. De FP staat voor FLAT-PACK en de 4 staat voor 200nsec.(logisch toch). Nou, FP-ic's zijn zo'n twee maal kleiner in alle richtingen dan de gewone ic's en zij worden bovendien niet met d'r pootjes door een print gestoken, maar bovenop de print gesoldeerd.
Daardoor passen er op dat printje met gemak 4 6116s aan iedere kant. Voor de decodering van de chip-select lijnen zit er dan ook nog een 74LS138 op de print. Dat is dan weer zo'n gewoon ic., en ik moet zeggen, die staat er wel onelegant bij, zo hoog op de poten.

Hoe het een en ander bedraadt is, zie je op onderstaand schema. Daarin staat ook hoe die module aangesloten kan worden op de ATOM (PL6/7). Voor de duidelijkheid : deze tekening geldt voor gebruik van die RAMs op de addressen #4000 tot #8000.(zonder battery-back-up). En de Addres- en Databusbuffers moeten zijn geplaatst.(ic 2 t/m 4)
Voor battery-back-up moet in plaats van de 74LS138 een 74LS156 worden gebruikt. (zie fig.2)
De laatste heeft een open-collector uitgang. Ook de NWDS moet een oc. krijgen en al die uitgangen natuurlijk ook weer een pull-up weerstand naar de + van de battery. Verder is nog een 74LS01 of vergelijkbaar nodig.
De makkelijkste realisering daarvan is : een set weerstanden op de plaats van de 74LS138 en een 74LS156 op een apart printje elders. De Aqu.print punt 25 komt dan aan de +batt..
Moet de module zijn dienst bewijzen van 0 tot #4000, dan is er iets meer nodig. Dan moet nl. alles wat daar nu zit verdwijnen. Alle "lage" 2114s (dat scheelt 33mA per stuk aan stroom) en de Floppy Disc Controler (die zit (indien aanwezig) op #0A00 tot 0A07).
Trek alles wat 2114 heet en niet rechts op de ATOMprint zit eruit (ic10 t/m19 +ic51 en 52). Verder moet ic6 eruit, ic5 punt2 t/m 6 aan de +5V en ic5 punt 12 losgenomen en ook aan de +5V.
Klaar is Kees, voor die niet floppybezitters in elk geval. De floppybezitters moeten ook nog zorgen dat als de FDC (8271) spreekt, de RAM z'n mond houdt. Dat kan zoals getekend in fig.3, maar misschien ook door de adressering van die FDC te veranderen. Zie daarvoor een ander artikel in dit blad. Door nu A14 aan te sluiten op de Aqu.print punt 30 ipv.punt 10 en punt 10 aan de +5V., is die RAM geadresseerd op 0 tot #4000.
Tenslotte zoals ik het nu heb zitten.
Vooropgezet: ik heb zo ongeveer niets meer standaard, maar wel club-compatible.

(in principe komt het op hetzelfde neer). Ik heb 4 16kRAM modules geplaatst. Een voor het schermgeheugen (#8000-#9FFF). Inderdaad 16k is tweemaal wat werkelijk nodig is, maar nu kan ik nog eens overschakelen op een tweede scherm.
De tweede zit van 0 tot#4000.
De derde zit van #4000 tot #8000.
En de vierde zit dubbel geparkeerd op #1000-#1FFF, #6000-#7FFF en op #A000--als #BFFF op nul staat.(EP.0).
Die vierde module heeft battery-backup. Deze heeft bovendien voorrang boven de andere op dezelfde adressen. Wil ik de andere RAM gebruiken dan schakelt dat om met #BFFF=#80. De WRITE-protect wordt uitgeschakeld met #BFFF=8. Na inschakeling van de voeding komt #BFFF automatisch op 0, zodat direkt de battery-backup-RAM voor komt (protected). Daarin staat dan ook het CX-systeem.
Daar komt nog bij dat de floppy-controler op #BF08-#BF17 zit. Dit werkt nog niet goed. Daarover in een ander artikel meer.
De adres-bus buffer (ic4) gebruik ik niet, omdat alles in de kast zit. De in- en uitgangen zijn met elkaar verbonden.
In fig.4 zie je hoe het een en ander is verbonden.

noot *

bij de firma ALLWAVE zijn de 16 K. modules gesignaleerd voor f 39,- gesignaleerd.

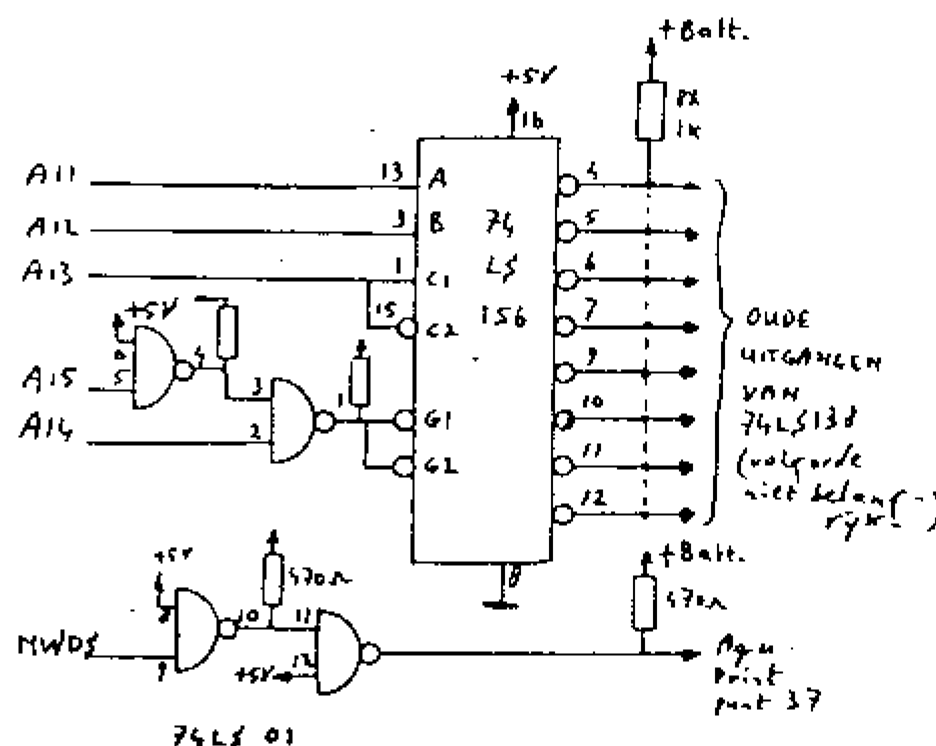


FIG. 2 : #4000-8000 + BATT. B.U.

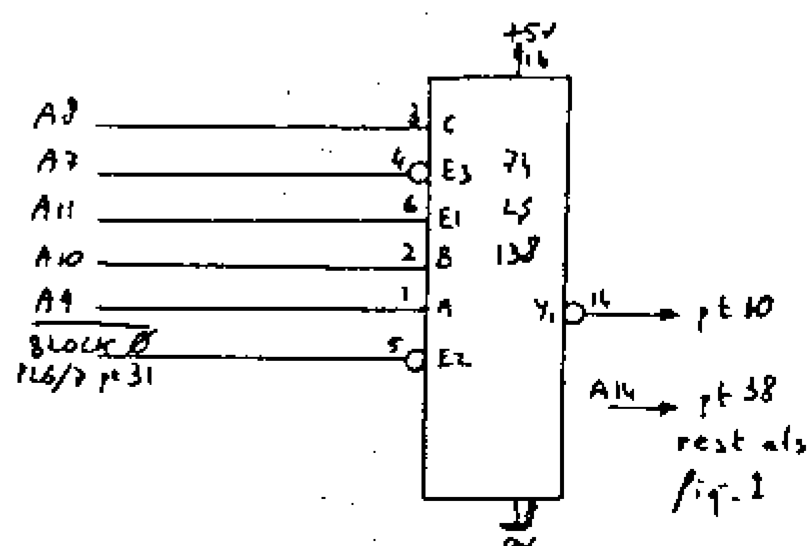


FIG. 3 : 0-#4000 min; #0AXX (FDC)
zonder Batt. BU.

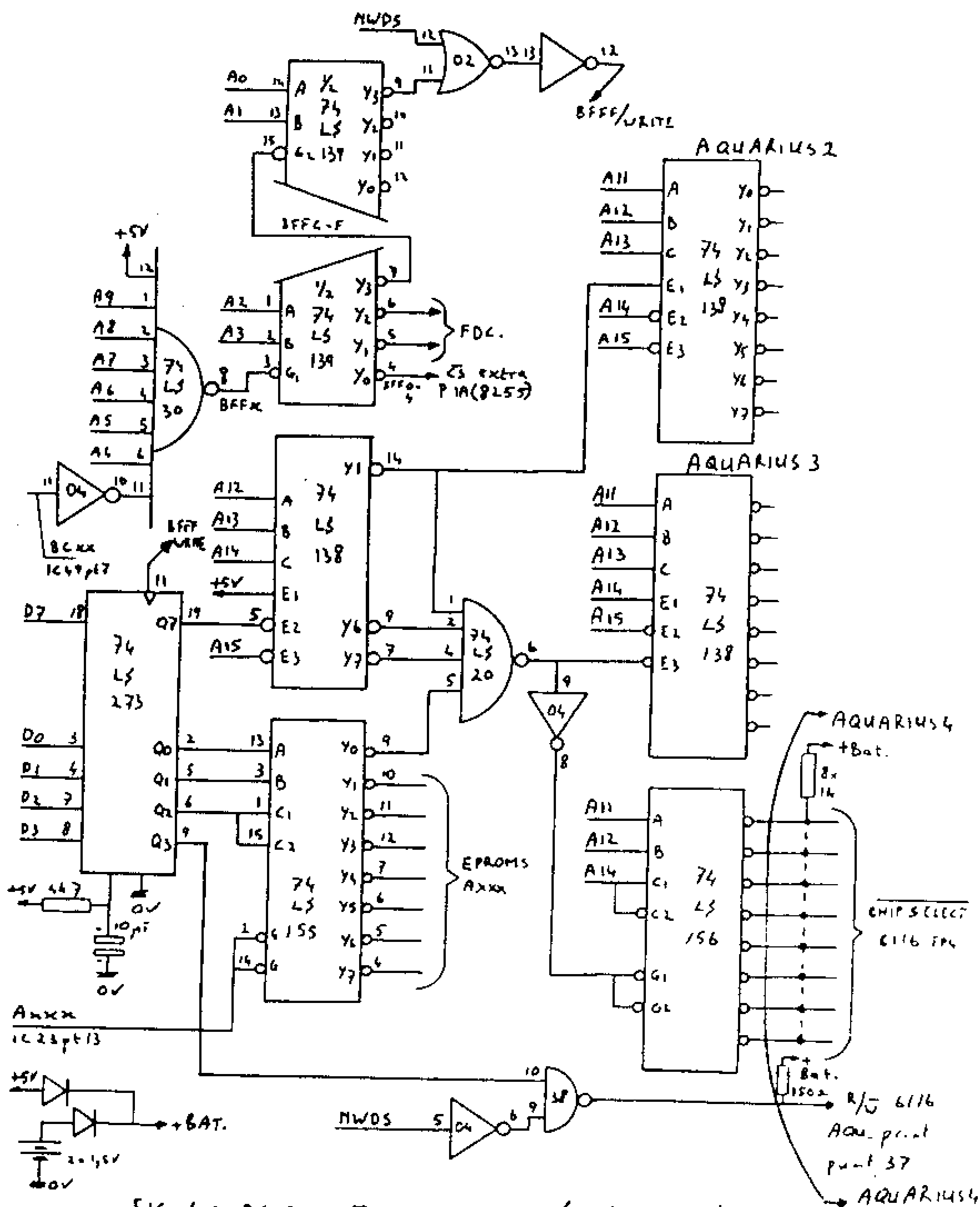
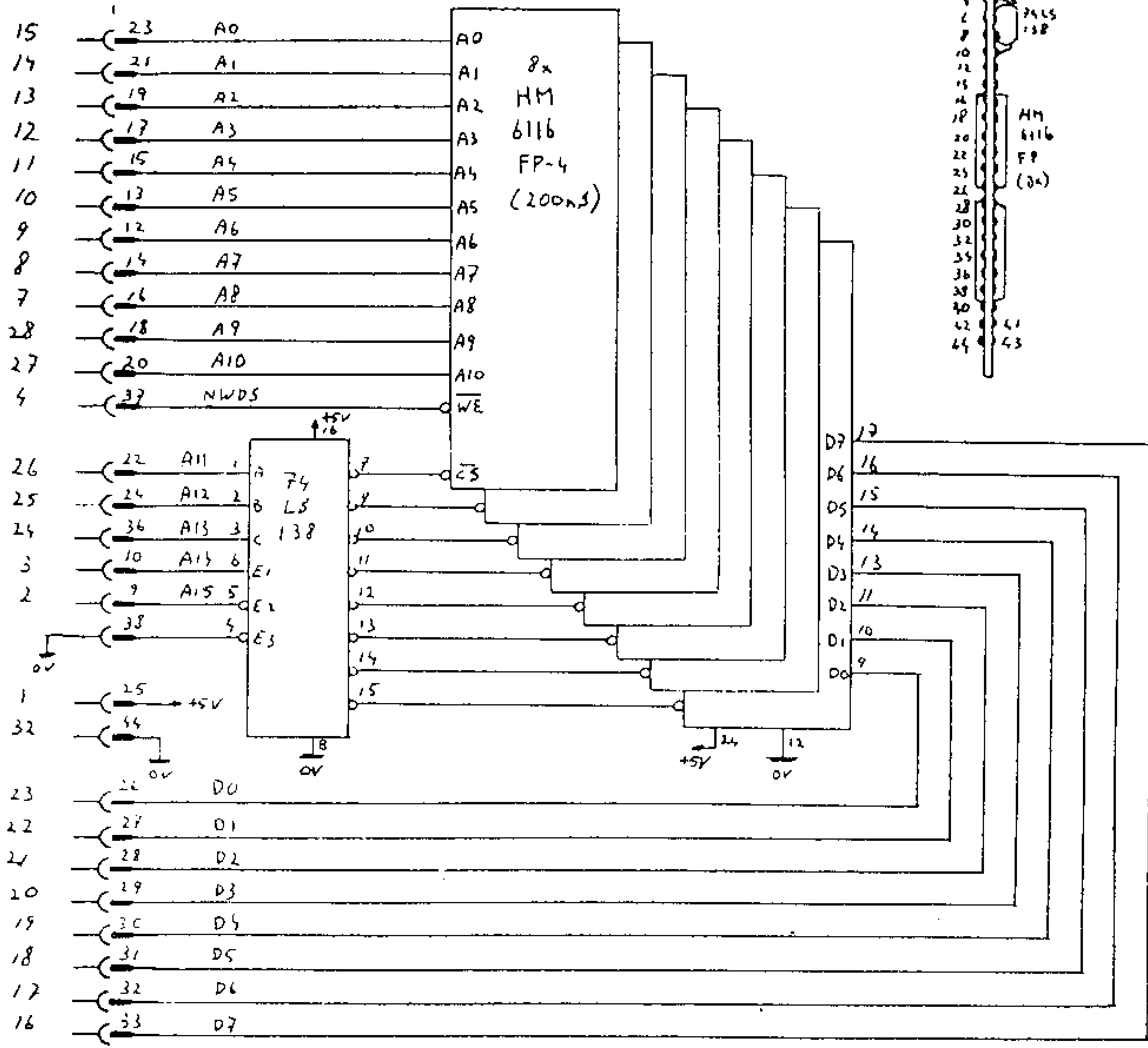


FIG. 4 : ACORN ATOM REIMERINK (gedeeltelijk)



PL6/7
AQUARIUS 16K MEMORY CARTRIGE
8x
HM
6116
FP-4
(200nS)
74
LS
138
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
NWDS
WE
CS
A11
A12
A13
A14
A15
D0
D1
D2
D3
D4
D5
D6
D7
+5V
0V
PIN nummers
HM
6116
FP

SPRITE- EN PAINTEDITOR

In Acorntjesbrood 2.4 heeft U alles kunnen lezen over GAGS, een fraaie toolkit van Gerrit Hillebrand, voor het maken van spelletjes en leuke grafische plaatjes.
Bij het maken van een spelletje zul je al gauw sprites willen gebruiken. Hiermee kun je gemakkelijk figuurtjes op het scherm zetten en laten bewegen. Een sprite moet eerst gedefinieerd worden voordat je hem kunt gebruiken. Dit kan op twee manieren: met het overzichtelijke, maar geheugen-vretende DEF-statement, of met het compacte CREATE statement. Bij CREATE moeten we steeds van binair naar hexadecimaal converteren en kunnen we (dus) moeilijk een sprite wijzigen. Toch zul je, zeker in de uiteindelijke versie van je programma, het CREATE-statement willen gebruiken, vanwege de compactheid ervan. Eigenlijk moeten we een gemakkelijke manier hebben om een sprite te maken en te wijzigen, terwijl we wel van het CREATE-statement gebruik maken.
Met de hier gepresenteerde SPRITE EDITOR is dit mogelijk; we kunen nu op een erg gebruikersvriendelijke manier een sprite maken en bewerken.
Om dit programma te kunnen runnen moet U P-Charme op het #1XXX gebied hebben gelocaliseerd, zoals dat is beschreven in Acorntjesbrood 2.4. Verder uiteraard de bijbehorende schakelsoft en GAGS-V2.0E. Misschien werkt het programma ook wel met P-Charme gewoon op #AXXX en goede schakelsóft (!!!), maar ik heb dat niet uit kunnen proberen. In elk geval moet U P-Charme en GAGS door elkaar heen kunnen gebruiken, dus GAGS-statements in procedures enz.
Voor de bediening van de sprite editor wordt gebruik gemaakt van een joystick.

SPRITE EDITOR

Na RUN kunnen we kiezen uit het aanmaken van een nieuwe sprite of het editen van een al bestaande. In dat laatste geval moeten we het BASE-adres van het gebied waarin de sprite is opgeslagen opgeven. Dan volgt een overzicht van alle bestaande sprites in dat geheugengebied. We zien achtereenvolgens het volgnummer van elke sprite, het assignmentnummer (is 0 als de sprite geen nummer heeft) en de naam van de sprite. Nu geven we het volgnummer van de te editen sprite op.
Dan wordt het eigenlijke plaatje (zie screendump) getekend waarin het bewerken van de sprites zich afspeelt. In het geval van het editen van een sprite verschijnt die sprite ook in beeld.
In het raster van 8*16 hokjes kunnen we een sprite invullen door de knipperende cursor met de joystick te bewegen. Met een druk op de knop van de joystick kunnen we een hokje inverteren. Rechtsonder zie je hoe de sprite er op ware grootte in het "echt" uitziet.
Rechts naast het hokjesraster staat een aantal commando's: SCROLL (in alle 4 richtingen), INVERT, TURN, ERASE en QUIT. We zetten de cursor (die er binnen het raster anders uitziet dan erbuiten) op het hokje voor het gewenste commando en drukken op de knop van de joystick. Het aangewezen commando wordt dan

uitgevoerd. Er verschijnt een horloge in beeld ten teken dat U hier even op moet wachten. Bij SCROLL moeten we de cursor op één van de vier pijltjes zetten. De sprite wordt dan 1 rij of kolom in de opgegeven richting verschoven (de vrijgekomen rij of kolom wordt met lege hokjes gevuld). INVERT inverteert de gehele sprite en TURN spiegelt een sprite om de verticale middenas. ERASE wist de hele sprite en met QUIT wordt de sprite editor verlaten. Bij ERASE en QUIT is een extra beveiliging ingebouwd tegen ongewenst uitvoeren. Na het geven van het commando verschijnt het woord OK (knipperend), en moeten we nog een keer op de knop van de joystick drukken. Pas dan wordt het commando uitgevoerd. Hebben we ons vergist, dan verplaatsen we de cursor gewoon in de een of andere richting, weg van het commando.
Het is niet mogelijk om de cursor naar posities op het beeldscherm te verplaatsen waar je niets te zoeken hebt.
Na QUIT kunnen we de naam en evt. het assignmentnummer van de sprite (als je dat in het CREATE-statement wilt hebben opgenomen) opgeven. De sprite editor vertelt U dan hoe het CREATE-statement moet luiden om de door U bedachte sprite te definiëren. Deze regel is nog NIET uitgevoerd, m.a.w. de sprite bestaat nog niet! Dus overschrijven, uitprinten of met de copy-toets kopiëren. Denk aan het goede BASE-adres!

PAINT EDITOR

Met het CREATE-statement is het ook mogelijk om patronen te definiëren voor het PAINT-statement van GAGS. Eigenlijk hebben we ook een PAINT EDITOR nodig om gemakkelijk patronen te maken en te wijzigen.
De werking van deze paint editor is analoog aan die van de sprite editor. Er zijn slechts een paar verschillen op te merken:

- Bij het editen krijgen we altijd 32 patronen te zien. (Nummer + 8 bytes definitie.)
- Naast het 8*8 raster zien we een vlak zoals dat er uitziet als we het met het huidige patroon inkleuren.
- Na QUIT krijgen we weer een CREATE-statement voorgeschoteld. Dit is al wel uitgevoerd! Het patroon bestaat dus al. Bij de sprite editor is het helaas niet mogelijk om het gegenereerde CREATE-statement ook uit te voeren; bij de paint editor wel.

Voor dat we de paint editor runnen moeten we het base-adres goed zetten (met het BASE-statement). Het patroon wordt gedefiniëerd op de base zoals die gold toen U RUN intoetste (bij 'create') of op de bij 'edit' opgegeven base.

De sprite editor staat van #2900 - #5A1E in het geheugen en de paint editor van #2900 - #55F5. Daarachter staat een aantal arrays en 2K sprite opslaggebied voor eigen sprites. Zorg er dus voor dat U voldoende geheugen aanwezig heeft.
Tot slot nog het volgende: Heeft U P-Charme niet op #1XXX, bekijk de listing van de sprite editor dan eens extra goed. Het onbeperkt kunnen mengen van P-Charme en GAGS is maar één van de vele voordelen die het plaatsen van P-Charme op #1XXX heeft. De bijbehorende schakelsoft is veel eenvoudiger en duidelijker dan overeenkomstige schakelsoft die uitgaat van P-Charme en GAGS op #AXXX naast elkaar.

GAGS SPRITE EDITOR

1234567890123456

SCROLL

INVERT
TURN

ERASE
QUIT

ACTUAL SPRITE IMAGE:

USE JOYSTICK TO DEFINE SPRITE

GAGS PAINT EDITOR

12345678

SCROLL

INVERT
TURN

ERASE
QUIT

ACTUAL PATTERN

USE JOYSTICK TO DEFINE PATTERN

DE NIEUWE MDCR-ROM

Korte beschrijving nieuwe software voor in EPROM/RAM op A-gebied. Deze software is upwards compatible met de vorige MDCR software. Dus oude file's kunnen weer teruggelezen worden etc.
In het hieronder staande overzicht is met een * aangegeven welke commando's niet gebruikt hoeven te worden als u volledig automatisch wilt werken, dus met een Header.
Alleen als u zonder Header wilt werken moet u gebruik blijven maken van sash en salo, in het andere geval is om te saven alleen SAM nodig.
In de hieronder getoonde voorbeelden kunt u zien dat de commando's in een basic programma opgenomen kunnen worden ook SAM commando's.
Voor de mensen die zelf basic programma's schrijven zijn hieronder een drietal mogelijkheden opgenomen.

1. Laden van een gewenste file kan ook als volgt:
   $#140="FILENAAM";LI.#A6C9
2. Saven van een file met de naam "FILENAAM" kan ook als volgt:
   $#140="FILENAAM"
   !#70=Beginadres
   !#72=Eindadres
   !#74=Startadres (zie ook bij salo).
   LI.#A6D7
3. Wilt u van een bepaalde filenaam het nummer weten waarmee het op de band staat dan als volgt:
   $#140="FILENAAM";LINK#A545;P.?#17F

Voor de mensen die zelf eenvoudige interface-schakelingen kunnen bouwen is ook nog het schema opgenomen. U heeft dus geen extra VIA nodig. Bij mij loopt dit nu bijna 2 jaar met op de A-poort een printer aangesloten.

HELP
Geeft een overzicht van alle commando's.

HELP	ini
sash *	DEL
salo *	LOCK
VERM	UNL
LAAD	CAT
REW	MHAND
wis	MAUTO
LRUN	UPD
SYN	FWD
MCAT	BVER
SAM	BACKUP

noot van de redactie:
de commando's met kleine letters dienen ook met kleine letters t worden gegeven. Dit als een beveiliging tegen fatale gevolgen, omdat nu de schift-toets ook gebruikt moet worden bij invoer.

Bij nogmaals op een toets drukken, worden een aantal voorbeelden getoond.

```
ini"BANDNAAM"
LAAD4;LAAD"FILE5"
FOR X=2TO5;LOCK X;N.
UNL"FILE3";UNL6       UNL=UNLOCK
DEL9;REW;wis;FWD      DEL=DELETE
LRUN(B+B)       LAAD EN RUN
SAM"FILE" 2900 3C00 29;P."BASIC"
SAM"PAINTER" 2800 3C00 2813
MHAND;F.X=3TO12;DELX;N.;MAUTO
UPD"FILE";UPD4            =UPDATEN
BVER  = BAND VERWISSELD
CAT
BACKUP = NAAR TAPE

    AMWUVS
```

Dan nu een overzicht van de commando's:

sash
Save met na het programma 40 miliseconde wissen.

salo
Save met na het programma 1 seconde wissen.
Alle programma's of datafile's die U op band zet krijgen een nummer en komen met een tussenruimte, die gewist moet zijn, op de band. Het eerste programma begint pas na 1 seconde (aanloopstuk). Dit wordt automatisch verzorgd. Doordat een programma altijd gevold moet worden door een stuk van 0,45 seconde gewiste band wordt normaal gesaved met salo.
Hierop is natuurlijk een uitzondering nl. als u een bestaand programma of datafile wilt overschrijven, dan wordt sash gebruikt. Duidelijk is wel dat er dan natuurlijk een file op band gezet moet worden die evengroot is als de vorige file. Een gesaved programma wordt automatisch gecontroleerd door het terug te lezen en de checksum te bepalen (verify).

```
salo"PROGNAAM" BBBB EEEE SSSS
                         OF SS
```

BBBB = Beginadres
EEEE = Eindadres
SSSS = Assembler startadres
  SS = Basic start adres met textpointer op SS
TOP en DIM wordt automatisch goed gezet.

VERM
Verify het programma met het aangegeven nummer X of "programmanaam". Haal het programma van band maar zet het niet in het geheugen, bepaal wel de checksum en controleer die met de op de band staande checksum.

LAAD
Laad programmanummer X of "programmanaam" naar het geheugen zoals door u op band gezet.

REW
Rewind de tape, deze is dan tevens gesynchroniseerd.

wis
Wis de tape vanaf deze positie tot aan het einde van de band.

LRUN
Laad programmanummer X of "prog.naam" en start het hierna op. Dit kan dus zowel in basic als in assembler zijn. Zie hiertoe bij salo en sash.

SYN
Synchroniseer het bandje. Dat wil zeggen positioneer juist en vul het juiste programmanummer in. Dit is het nummer waar de MDCR voor staat.

MCAT
Laat een catalogus zien van wat er op band staat. De band moet natuurlijk gesynchroniseerd zijn. Dit commando is te onderbreken met de CTRL-toets. Wel even vasthouden. Het bandje bijft dan gesynchroniseerd achter.

SAM
SAVE AUTOMATISCH MDCR.
SAM"FILE" 2900 3C00 29 (voor basic).
SAM"PAINTER" 2800 3C00 2813 (voor assembler).
Het programma wordt gesaved en de Header wordt automatisch bijgewerkt. U hoeft zich geen zorg te maken over kort of lang saven. Dit wordt automatisch verzorgd.

ini
ini"BAND 1A", hierna wordt gevraagd "Zeker weten j/n"
Bij "ja" wordt het bandje van voor tot achter gewist, en vooraan wordt hierna de Header aangemaakt van #2400 tot #2800

DEL
Delete X of "prog.naam". In de Header verschijnt dan een ruit-tekentje tussen het programmanummer en de programmanaam. Let wel het programma staat nog steeds op band dit wordt pas overschreven bij een volgende save, die exact dezelfde programma lengte heeft.

LOCK
Lock X of "prog.naam" Het programma wordt gelockt (beschermd). Gelockte file kunnen niet geupdate of gedelete worden. Tussen programmanummer en naam verschijnt het "#" teken.

UNL
UnlockX of "prog.naam". Het # teken verdwijnt en het programma kan worden geupdate of delete.

CAT
U krijgt een overzicht van de op de band staande programma's, tevens het totaal aantal gebruikte bytes, het aantal keren dat reeds gesaved is en het nummer waar de MDCR voor staat. Mocht de Header in het geheugen "beschadigd" zijn dan wordt deze opnieuw van band geladen.

MHAND
Met dit commando wordt het MDCR-systeem op handbediening gezet. Na iedere transactie wordt de Header bijgewerkt, maar nog niet op de band geschreven. Makkelijk als meerdere programma's gesaved of bijv. gedelete moeten worden, na de laatste transactie en voordat u de computer uitzet voert u in MAUTO en de Header wordt automatisch op de band bijgewerkt.

MAUTO
Zet het MDCR-systeem op automatisch en saved de Header zoals die nu is.

UPD
Commando om een bestaand programma of file te updaten. Let wel het programma of file kan niet groter zijn als het origineel programma. Dus houdt er rekening mee als het een program is waaraan nog geknutseld moet worden, en u wilt het over de vorige versie heenleggen save dan met meer ruimte. Overschrijven van het volgende programma is niet mogelijk. (Is tegen beveiligd in de soft).

FWD
Laat de tape vooruit lopen. Is te stoppen met ESC-toets.
De synchronisatie wordt automatisch verstoord.

BVER
Bij iedere bandverwisseling te gebruiken, als u automatisch wilt werken.
De Header wordt geladen en er verschijnt een overzicht zoals genoemd onder CAT.

BACKUP
Backup van MDCR naar normale cassette recorder. Kan zowel met 300 als 1200 Baud. Bij het commando BACKUP wordt het eerste op de band staande programma geladen en er verschijnt
RECORD TAPE
Zet de bandrecorder op opnemen en druk op een toets. Het programma wordt nu op cassette gesaved. Daarna laadt de MDCR het tweede en volgende programma's en deze worden weer gesaved. (Volledig automatisch).

FOUTMELDINGEN
============

101 Geen cassette aanwezig.
102 Begin of einde van de band
103 Niets op de band.
104 Preamble niet ontvangen.
105 Niet correct geladen
106 Schrijven niet toegestaan.
107 Bij controle van het programmanummer op de band met het door de software aan de hand van de Header berekende programmanummer is een verschil geconstateerd. Dit kan bijv. optreden als u gaat saven terwijl u een ander bandje in de MDCR heeft gedaan zonder BVER te doen.

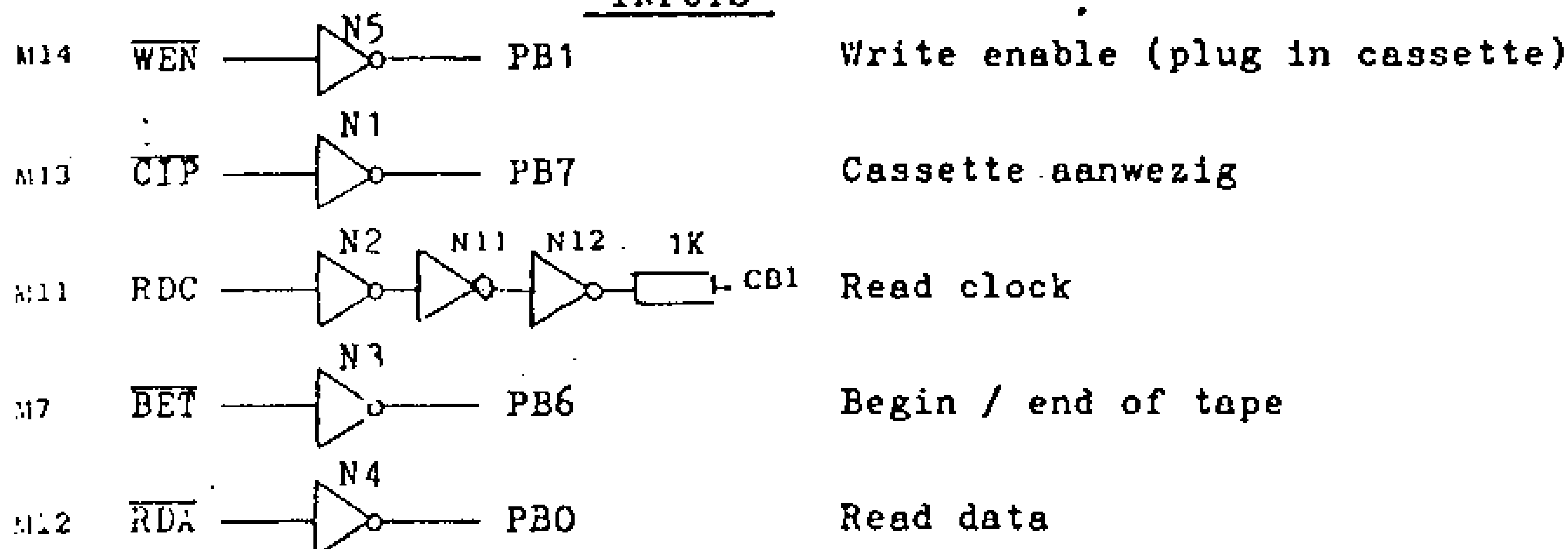
De werking van de MDCR is te onderbreken door de escape-toets vast te houden, of door de break-toets in te drukken. In het laatste geval wordt de zaak altijd direct afgebroken.Voor de liefhebbers er is nog wat ruimte over, als u de soft in EPROM zet.

VEEL SUCCES

MDCR van PHILIPS

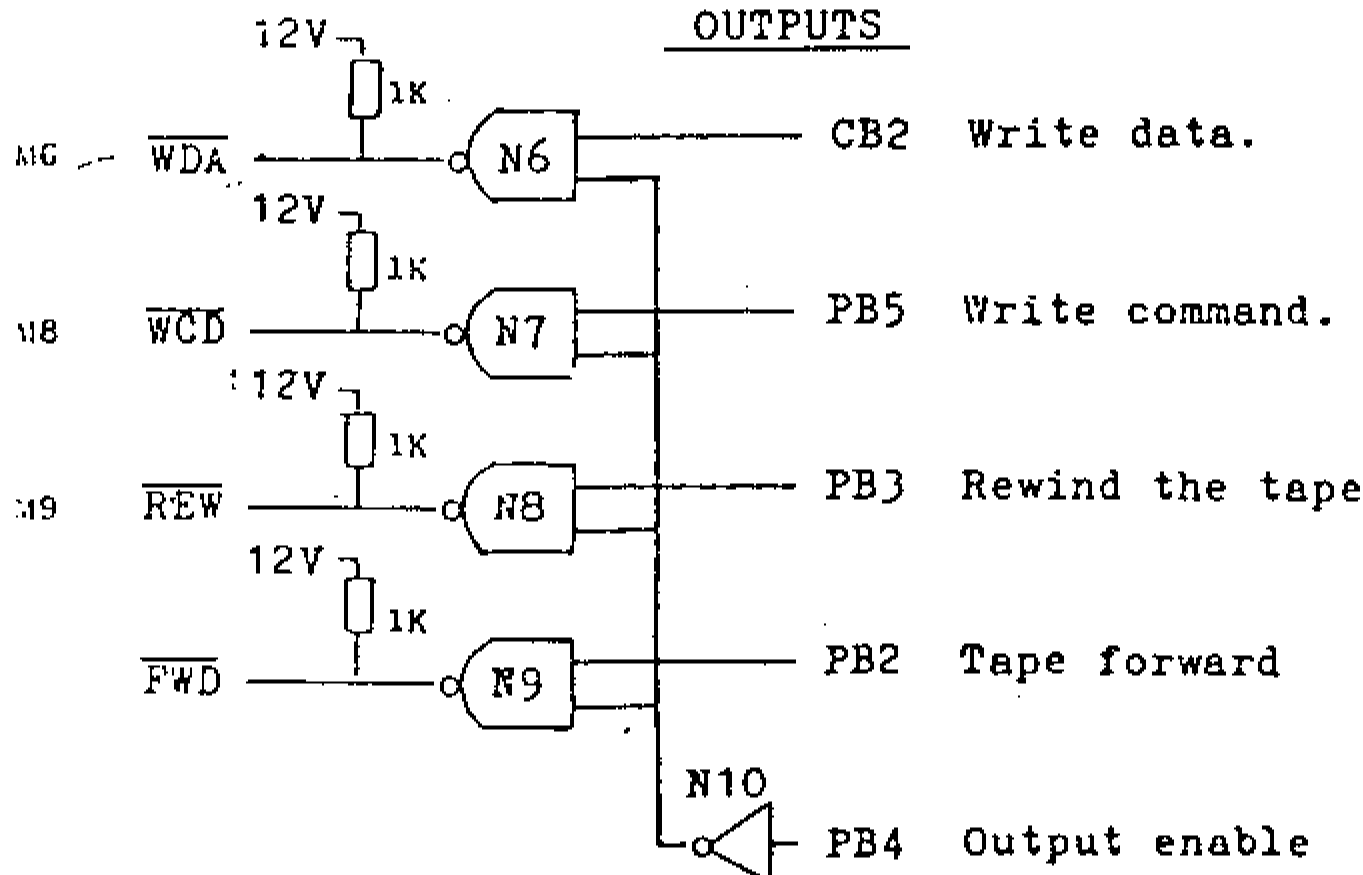
Interface schakeling die op de B-poort van de ACORN ATOM moet worden aangesloten.

INPUTS



Al deze input signalen worden geinverteerd en aangepast door IC1 (CD4049 N1 - N5)

OUTPUTS



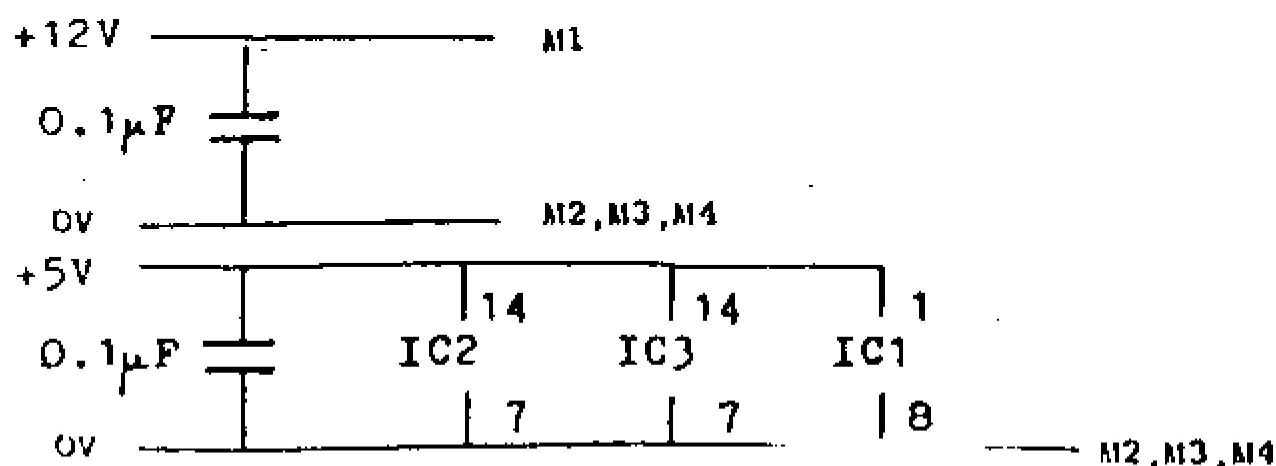
IC1 N1-N5 CD4049

IC2 N6-N9 74LS26

IC3 N10-N12 74LS04

M1 tot en met M14 zyn de MDCR interfacenummers

+5V pen 1 , OV pen 32 van de ACORNBUS



SALFAA

NOT(uitleg van MAC/BP)

Inleiding

Voor de vakantie (in Acorntjesbrood 2.3) heb ik een assembler aangekondigd die ik inmiddels MAC/BP heb gedoopt, hetgeen staat voor: Making Assembly Code by Basic Programming, door sommigen echter reeds vertaald door Macro assembler van Bram Poot. Dit laatste komt echter voor hun rekening. De essentie van MAC/BP is dat de assemblerinstructies in feite basic-statements zijn geworden. Dit biedt dan de mogelijkheid om de faciliteiten van bv. P-charme los te laten op assemblerprogrammering. In het bij de aankondiging afgedrukte voorbeeld wordt de functie BIN in 'n assembleradresseringsmode gebruikt en de procedure INR fungeert er als macro.
Het nadeel van MAC/BP, tevens de feitelijke reden dat ik de reeds min of meer werkende versie niet verder ontwikkel,is dat, omdat er geen verschil meer is tussen assembler en basic, het ook niet meer duidelijk is wanneer wel en wanneer niet iets afgedrukt moet worden. Dit is natuurlijk wel met P.$6 en P.$21 te doorbreken, maar, en in de praktijk bleek dat ook wel, als je een mooie assemblerlisting wilt creëren (dus datgene dat tijdens vertalen wordt afgedrukt) vraagt dit zoveel overhead dat de lol van het assemblerprogrammeren er snel afgaat.
De assembler die na MAC/BP min of meer uit dezelve is ontstaan, heb ik SALFAA genoemd, hetgeen staat voor: Symbolic Assembly Language For Acorn Atom, een naam die ook aan mijn eerste assembler gegeven had kunnen worden ware het niet dat ik dat destijds eenvoudigweg niet heb gedaan.

Samenvatting

Wat valt er, zonder direct in details te treden, over SALFAA te zeggen?
SALFAA is symbolisch, hoort als utility op het AXXX-blok en kan als zodanig als losstaande kit gebruikt worden (MAC/BP heeft P-charme nodig). Via een optie kan echter P-charme in principe dienstbaar gemaakt worden aan SALFAA (denk aan FUNCTION, NOT, etc.).
SALFAA is mengbaar met basic, zoals u inmiddels wel zult hebben begrepen,maar het basicgedeelte kan zo klein uitgevoerd worden, dat u eigenlijk een echte assembler onder de vingers heeft.
SALFAA zal binnenkort SWEET16-code kunnen genereren. Natuurlijk moet u, om dit uit te buiten, ergens een SWEET16-interpreter hebben staan (zie eerdere uitgaven van Acorntjesbrood).
SALFAA heeft de beschikking over in principe 26 directives; u raadt het al: de letters van het alfabet (enkele hiervan zijn (nog) niet geimplementeerd).

Overhead

Laten we SALFAA dan nu wat nader bekijken. Het geheel is een kant en klaar pakket dat de volgende opdrachten kent: ASM-, PASS, CALL, (, en \.
De back slash (\) wordt gebruikt voor commentaar totaan een ; of CR, zowel in het basic- als assemblergedeelte. De ronde haak open (() wordt gevolgd door een willekeurig karakter waarna alles, regelnummers incluis, als commentaar wordt opgevat totaan datzelfde karakter gevolgd door de ronde haak sluiten ()) (zie ook het bij de aankondiging van MAC/BP afgedrukte voorbeeld). Ook dit geldt zowel in basic- als assemblermode.
CALL doet bijna hetzelfde als LINK, maar nu wordt behalve A, X en Y ook het processor status byte gevuld en wel met de inhoud van het laagste byte van de variabele P. Bovendien mag als argument ook een symbool uit de symbol table worden ingevuld, bv. CALL ENTRY. Een expressie is eveneens mogelijk: CALL ENTRY + 3. Om de een of andere dubieuze reden blijkt CALL voorlopig alleen in direct mode goed te werken.
PASS neemt de laagste twee bits van de erop volgende expressie en plaatst deze in het assembler status byte (hierover later meer).
ASM- tenslotte verwacht direct achter het min-teken een van de letters B,C,I,S,U,V waarna alle karakters daarna totaan ; of CR als commentaar opgevat worden, zodat u dit soort constructies kunt maken: ASM-BEGIN OF ASSEMBLY of ASM-CONTINUE PART II;.
De letters hebben de volgende betekenis:

```
B - BEGIN     de symbol table wordt geinitieerd (op #8200) var S
              de object code pointer wordt ,,   (op #9000) var P
              de memory pointer wordt      ,,   (op #0000) var O
              een nieuwe break handler wordt gezet en de oude
                wordt bewaard
              het scherm wordt afgezet
C - CONTINUE een nieuwe break handler wordt gezet en de oude
                wordt bewaard
I - INITIATE de symbol table wordt geinitieerd (op #8200) var S
              de object code pointer wordt ,,   (op #9000) var P
              de memory pointer wordt      ,,   (op #0000) var O
              het scherm wordt afgezet
S - SYMBOL    de symbol table wordt afgedrukt
U - UNRAVEL   de symbol table wordt gesorteerd
V - VERSION   het versienummer wordt afgedrukt
```

(deze laatste drie worden nader besproken bij de directives. Het zijn de basic-equivalenten van .S, .U en .V)
ASM-B en ASM-C veranderen het programma naar assemblermode, bij de overige blijft het in basicmode.

Structureel

Voordat ik verder ga met SALFAA, wil ik eerst een structureel aspect bespreken.
Aangezien P-charme zo ongeveer de enige echte Acorn Atom basic upgrade kit vormt, zat het me niet lekker om deze naast de "gewone" boxen op het A-blok te zetten. Als ik P-charme ergens anders zou plaatsen, kon ik:

a. gebruik maken van de P-charme routines die dan nooit weggeschakeld worden.
b. via schakelsoft P-charme met een andere box combineren (bv. CALC en 1200 baud, GAGS-routines in procedures etc.).
c. kortom uitbreidingen die je niet doorlopend, maar wel vaak, nodig hebt (bv. assembler, extra commando's (let wel: geen statements) ed.), naast elkaar i.p.v. achter elkaar zetten.

Dit nu is bij mij verwezenlijkt. Ik heb P-charme gerelocaliseerd naar het #1XXX-blok (versienummer 1). Dat dit niet zonder meer gelukt is, zal eenieder wel duidelijk zijn. Ik heb drie concessies moeten doen:

1. het executieadres van BSAVE is nog steeds #AFAF, hoewel daar geen P-charme meer staat. Als ik het zou aanpassen werd het #1F1F en daar staat een andere routine. Aangezien ik echter zelden basicprogramma's *RUN, vind ik dit niet zo erg. Bovendien kan ik i.p.v. BSAVE altijd nog *SAVE"..."..... .... 1FAF intoetsen.
2. RENUM is RENU> geworden (het hoe en waarom hiervan zou me 'n stuk of wat kantjes kosten aan uitleg). Aangezien ik echter meestal REN. intik, gaat dit zonder meer goed. Bovendien krijg ik nu via RENUM (en uiteraard schakelsoft) de supersnelle Josbox-RENUM (als ik toch geen computed GOTOs, GOSUBs en/of RESTOREs heb, gebruik ik deze meestal).
3. in AUTO heb ik een instuctie moeten schrappen, ter verkrijging van twee broodnodige extra bytes. Deze ingreep blijkt tot op heden geen gevolgen gehad te hebben (?????).

Het nadeel is dat ik routines die naar P-charme linken (zoals de .STAT routines) niet domweg kan overnemen. Als ze met een referentie zijn geprogrammeerd, valt het aanpassen nog wel mee, maar helaas is dat zelden het geval.
In probleem- en testgevallen kan ik echter altijd de originele P-charme in het A-blok inladen en die op #1XXX disabelen. De basicprogramma's leveren hoe dan ook nooit problemen op, omdat ik ervoor gezorgd heb dat de basiccompatibiliteit behouden is gebleven. De enige uitbreiding achter P-charme is de schakelsoft ter lengte van maximaal 256 bytes. Alle andere uitbreidingen kunnen dan naast elkaar in de schakelkaart.
Als ik in wat verder volgt het over P-charme heb (zoals gezegd niet vereist bij SALFAA), dan is dat met betrekking tot deze veranderde versie.
Gaan wij nu weer verder met SALFAA.

Status byte

Laat ik dan eerst het assembler status byte beschouwen.
Ik heb daarvoor #28FF gekozen (wordt op nul gezet door PROGRAM)
De bits hebben de volgende betekenis:

bit	0	/	1	
0	eerste	/	laatste	pass
1	wel	/	niet	wegschrijven
2	xxx	/	xxx	gereserveerd
3		/		
4	64+	/	32	karakters per regel
5	geen	/	wel	page cross melding
6	wel	/	geen	P-charme
7		/		

Een aantal zaken is pass-afhankelijk: OUT OF RANGE & UNDECLARED SYMBOL zijn pas foutmeldingen tijdens de tweede pass. REDEFINED SYMBOL is een waarschuwing tijdens de eerste pass. Dit wordt gerelateerd aan bit0. Als bit1 hoog is wordt de code niet daadwerkelijk naar geheugen weggeschreven. Laag -> wel. Deze faciliteit bleek later handig bij het ontwikkelen van schakelsoft. Deze beide bits worden door het PASS-statement beinvloed (zie aldaar). De overige bits worden beinvloed door het .O (option) directive (zie aldaar).
Bit2 is gereserveerd voor de toekomst. U kunt dit 0 of 1 maken, evenals de twee niet ingevulde bits, die (nog) geen toepassing hebben. Bit4 zet u het beste laag bij gebruik van 80-kolomsVDU en evt. hoog bij de normale AtomVDU. Bit5 geeft aan of de branches over een paginagrens gemeld moeten worden en bit6 geeft aan of er ja dan nee een P-charme aanwezig is.

Directives
~~~~~~~~~~

Een directive bestaat uit een punt, onmiddellijk gevolgd door een symbool waarvan alleen de eerste letter significant is. U kunt dus schrijven:
.BYTE (is duidelijk), .B (compact) of .BOTERBABBELAAR (raar).
We hebben op deze manier 26 directives (de eigenlijke betekenissen zijn voor de duidelijkheid ingevuld):

.ASCII \string\ Na ASCII komt een willekeurig niet-spatie karakter (gebruik daar liever niet de slash (/) voor, omdat u die met P-charme niet terug kunt vinden). Alles tussen twee van deze delimiters wordt als string opgevat en de asciicodes van de karakters worden in het geheugen gepoked. Na de tweede delimiter mag u een komma plaatsen waarna bytes worden gepoked. Bv. .ASCII\STRING\,#0D,#0A

.BYTE leest een of meer expressies en plaatst de laagste bytes ervan achtereenvolgens in het geheugen.
Bv. .BYTE TOP, 5*7,0, WAARDE+1

.CODE leest een adres en plaatst dit in de variabele P die ook bij SALFAA als object code pointer fungeert.Deze wordt door ASM-B op #9000 gezet.

.DBYTE plaatst een 2-bytes adres in het geheugen, het high byte eerst. Bv. .DBYTE adres (ofwel .BYTE adres/256,adres%256).

.END zet de printer uit, het scherm aan, de breakvector terug en verandert het programma naar basic-mode. Achter .END mag commentaar staan totaan ; of CR. Bv. .END OF ASSEMBLY;

.FIRST forceert een listing tijdens pass 0.

.HARD zet tijdens pass 1 de printer aan t.b.v. een hard copy. Als de printer in off line mode staat of niet aanwezig is, blijft 't programma niet domweg hangen,maar meldt dit feit. Het omschakelen naar on line danwel het indrukken van de shifttoets continueert het programma.

.IF is qua syntax analoog aan de normale IF (zonder THEN).

.LIST Normaal worden beide passes niet gelist. .LIST zet tijdens pass 1 het scherm aan.

.NEWLINE wordt gevolgd door een 1-byte expressie. Evenzovele linefeeds worden in de listing ingelast.

.OPTION beinvloedt de hoogste 6 bits van het status byte. De laagste 2 bits blijven onveranderd.
Bv. .OPTION #40 -> P-charme niet aanwezig.
~~~~~~~~~~

.PRINT leest een of meer expressies en stuurt deze tijdens pass 1 naar de printer driver (dus niet ook naar het scherm).

.RAM leest een adres en plaatst dit in de variabele O. Als deze waarde ongelijk is aan de defaultwaarde O (zie ook ASM-), dan wordt de vertaalde code niet naar P, maar naar O weggeschreven.
Als u bv. een toolkit wilt schrijven, is .CODE #A000 niet voldoende, omdat daar SALFAA (in EPROM?) staat. In zo'n geval schrijft u .CODE #A000 .RAM #9000 Na vertaling schakelt u RAM in bv. ?#BFFF=0, en toetst in COPY #9000,#9FFF,#A000.

.SYMBOL drukt de symbol table af. U kunt de listing stoppen en weer doorstarten met de REPTtoets. Na de tabel wordt u nog verteld hoeveel waarschuwingen er zijn opgetreden.

.TABLE leest 1 of 2 adressen.
Geeft u 2 adressen, bv. .TABLE #8400,#8800 dan reserveert u 1k bytes voor de symbol table vanaf #8400. Beide adressen worden in de variabele S bewaard.
Geeft u 1 adres, bv. .TABLE #9700 dan bepaalt SALFAA zelf het eindadres door het eerste niet-RAM adres op te zoeken, waarbij data niet verziekt wordt. Zorg er echter wel voor dat SALFAA niet in RAM staat als u #9800 en hoger hebt gestapeld! Ook nu worden beide adressen in S bewaard.
De default adressen zijn #8200 (begin) en #9000 (eind).

.UNRAVEL Ontrafel (sorteer) de symbol table. Ik geef toe dat deze naam ietwat gezocht is. Het was echter een van de weinige letters die overbleven.Onder het motto "graag of niet" heb ik gemeend het zo maar te laten. Als sorteermethode heb ik gekozen voor de o zo ineffectieve, maar o zo eenvoudig te programmeren bubblesort. Onder hetzelfde motto "graag of niet" leek me dit wel gerechtvaardigd. Bedenk overigens wel dat de symbolen niet allemaal evenlang zijn, hetgeen voor andere dan bubblesort-methodes extra problemen geeft. Het sorteren gaat nu uiteraard wel traag: een beetje programma dat ca. 1 minuut vertaalt, sorteert grofweg eveneens in een minuut (als de tabel tenminste flink door elkaar ligt).

.VERSION drukt het versienummer van SALFAA af. Het is een goede gewoonte om in uw programma te vermelden voor welke versie u het geschreven heeft. Makkelijk voor anderen, maar zeker ook voor uzelf.

.WORD plaatst een 2-bytes adres in het geheugen, low byte eerst.

Symbolen

Assembler labels worden op de gebruikelijke manier gedefinieerd met de dubbele punt, dus :SYMBOOLNAAM. De naam wordt in de symbol table bijgeschreven of als deze al bestond overschreven met daarbij de waarde die P op dat moment heeft. Als de symboolnaam wordt gevolgd door "=" dan wordt i.p.v. P de waarde van de achter het gelijk-teken staande expressie genomen.
Voorbeeld:

```
  :ZERO   = 0
  :WAARDE = #FF
:FILL LDY@ZERO
:LOOP LDA@WAARDE
      STA #8000,Y
      INY
      BNE LOOP
      RTS
```

Symbolen mogen uit maximaal 26 letters bestaan, met dezelfde beperkingen als bij MINIAS. Als u echter altijd kleine letters gebruikt (daar niet, hier wel mogelijk) kan er nooit iets misgaan. Als leesbaarheidskarakter is de single quote (') gehandhaafd. Dus DIT'IS'EEN'SYMBOOL is inderdaad een symbool.
Foutmeldingen treden op als een symbool te lang is, als een illegaal karakter wordt gevonden of als het symbool niet in de symbol table past.

DOS- en COS-commando's
~~~~~~~~~~~~~~~~~~~~~~

Stercommando's mogen in SALFAA voorkomen. Handig voor het inladen van reeds gecreeerde of standaard tabellen.
Bv. *LOAD"TABLE"8200
Bedenk wel dat dit zowel in pass 0 als pass 1 gebeurt. Gebruik eventueel constructies als:
.IF ?#28FF&1=0; *LOAD"TABLE" of
.IF EVEN(?#28FF); *LOAD"TABLE"

Listing
~~~~~~~

De spaties voorafgaand aan een instructie worden in de listing genegeerd, waarna er bij directives 1 wordt ingevoegd, en bij instructies 2, zodat de dubbele punten van labels onder elkaar komen,de punten van directives onder elkaar 1 positie verder en de instructies ook onder elkaar wederom 1 positie verder.
Als u meerdere instructies op 1 regel plaatst, worden ze, vanaf de instructie die in behandeling is, allemaal afgedrukt. Heeft u bv. deze regel :INI LDY@0; STY#80 dan drukt de listing af:

```
:INI LDY@0;    STY#80
  LDY@0;    STY#80
   STY#80
```

Toekomst
~~~~~~~~

De bron van SALFAA (geschreven in symbolische assembler) loopt van #2900 tot #8000. Deze levert me versie 0.4, waarmee ik de volgende versie van SALFAA in SALFAA ga schrijven.Daarin zullen storende fouten verbeterd zijn, SWEET16-instructies herkend worden en datgene waar nog behoefte aan was geimplementeerd zijn.

Verantwoording
~~~~~~~~~~~~~~

SALFAA is ontstaan na enig brainstormen tussen Frans van Hoesel, schrijver van MINIAS en mijzelve. schrijver van MAC/BP. MINIAS is geschreven op compactheid, MAC/BP is geschreven op mogelijkheden. Frans heeft mij ervan overtuigd dat MAC/BP toch eigenlijk wel te lang is (2.5 k). In praktijk zul je iets dergelijks niet permanent aanwezig hebben en ook niet telkens willen hoeven inladen. Aan de andere kant wilde ik een aantal faciliteiten niet kwijt (bv. variabele O als memory-pointer, onmogelijk in MINIAS). De oplossing lag voor de hand: de schakelkaart. De 3.5 k van SALFAA zijn nu pas aanwezig als ze nodig zijn.
Ik besef terdege dat deze handelswijze niet voor iedereen interessant is. Vereisten zijn:

- P-charme op #1XXX (zie ook "structureel")
- schakelkaart
- goede schakelsoft
- gecharmeerd zijn van mooie assemblerlistings, zowel voor als tijdens vertalen
- het zien van andere voordelen:
  - sprites in procedures
  - functions in GRMOD
  - textfiles van 1200 baud
  - external libraries op A-blok
  - alles wat hier nog niet te overzien valt (en dat is veel!)

Bent u een verwoed machinetaalprogrammeur en bent u gevoelig voor mijn filosofieën, dan voorspel ik u vele genoeglijke uurtjes met SALFAA. Voldoet u alleen aan het eerste, ook dan kunt u, zij het in wat mindere mate, plezier beleven aan SALFAA.

Voorbeeldprogramma voor SALFAA

Het hierbij afgedrukte programma SWITCH SYSTEM laat enkele zaken zien die met SALFAA mogelijk zijn. De aanhef van het programma is wat ingewikkeld, maar dat komt omdat de schakelsoft wordt aangemaakt die eigenlijk al nodig is om naar SALFAA door te sturen. Als er nog geen schakelsysteem aanwezig is, moet de back slash in regel 20 worden weggelaten. Is dat systeem er wel dan mag het programma zijn zoals het is.
In mijn systeem zit P-charme op #1XXX, zoals te zien is aan enkele sprongadressen (r.370,1100). P-charme kan gedisabled worden via het hoogste bit van #BFFF. Op dit moment zit SALFAA (versie 0.4) in een testuitvoering in voet 0 van de schakelkaart. Ik denk dat de SALFAA-aspecten wel voor zichzelf zullen spreken. Zo niet, schroom dan niet om opheldering te vragen.
Voor de volledigheid wil ik nog wat schakel-aspecten bespreken die echter niets te maken hebben met SALFAA en er dus volledig los van staan.
Het schaduwbyte heb ik in #B80A geplaatst. Dit is het shiftregister van de VIA (moet je dus wel hebben!) en een van de weinige bytes die echt absoluut door niemand (dacht ik) gebruikt worden. Dat is dus bij deze verleden tijd geworden. Het moet uiteraard ooit een keer veranderd worden, maar voorlopig red ik me er prima mee.
ROM (r.270) wordt gebruikt om te schakelen en - (r.290) is handig om array-elementen van een programma op te vragen bv. -P.RR(2,3). Dit is dus hetzelfde als p dat ooit als p.STAT in AcornNieuws is beschreven.

Aspecten van de schakelsoft zijn:
1. aangezien het schakelen via de breakhandler verloopt, wordt er niet geschakeld als BRKVEC op een "vreemde" waarde staat.
2. er wordt niet geschakeld als een van de vectoren NMIVEC, BRKVEC, IRQVEC, COMVEC, WRCVEC, of RDCVEC naar het A-blok wijst. Let er dan wel op dat na bv. FCOS de Josbox ingeschakeld blijft of wordt op het moment dat er een COS-operatie gepleegd wordt(of gebruik in dit geval alleen COS0 en COS1).
3. de reeds ingeschakelde box wordt als eerste doorlopen i.v.m. o.a.grotere verwerkingssnelheid. Hiertoe wordt bit6 gebruikt (r.660). Daarna van 0 tot 7 de andere boxen.

```
 10 REM SWITCH SYSTEM / SALFAA V0.4
 20\?#BFFF=#80;REM P-charme off / SALFAA on
 30 PASS 2;GOSUB c;REM don't write to memory
 40 PASS 1;GOSUB c
 50 ?#BFFF=#00;REM P-charme on
 60 END
 70cASM-BEGIN ASSEMBLY OF SWITCH SYSTEM
 80 .LIST .HARD .OPTION #40
 90 .PRINT 15                  \ condensed mode
100 .CODE #3FC                 \ extension byte
110 .BYTE #FF                  \ don't extend
120:BASE = #2400               \ or any other RAM page
130 \ DECLARATIONS:
140:SWITCH   = #B80A           \ or any other RAM byte
150:BRKVEC   = #202
160:ORGBRK   = #C9DB           \ original break handler
170:LATCH    = #BFFF
180:TEMP     = #4F             \ switch/no switch flag
190:ROMNUM   =  8              \ number of eproms
200:ERROR'94 =  94
210:LF       = #0A
220:CR       = #0D
230:NOP      = #EA
240
250 .CODE BASE
260 .BYTE #FF,#E3,#C6
270 .ASCII"ROM"
280 .DBYTE   ROM;#B000
290 .ASCII"-"
300 .DBYTE DIRMO;#8000
310 .DBYTE ENTRY;#8000
320 .BYTE #80                  \ end of 'table'
330:ROM
340 JSR #C4E1; DEC #04; LDX #04; LDA #16,X
350 STA SWITCH; STA LATCH; JSR TEST; JMP #C55B
360:DIRMD
370 JSR #142E; JMP #C31B
380:ENTRY
390 LDA BRKVEC + 1
400 (-----------------------------------------------
410  For some very silly reason it is necessary to
420  enclose a symbol in brackets in this kind of
430  addressing modes:
440 -----------------------------------------------)
450 CMP@(BREAK)/256            \ try to switch if BRKVEC
460 BEQ TRY'SWITCH             \  has been set to BREAK
470 CMP@(ORGBRK)/256           \ has BRKVEC been re-set?
480 BNE DON'T'SWITCH           \ yes, don't switch
490:TRY'SWITCH
500 LDX@#08                    \ check if any vector
510:LOOP                       \  points to AXXX
520 LDA #200,X                 \ get high byte of vector
530 AND@#F0                    \ mask low nibble
540 CMP@#A0                    \ vector to AXXX?
550 BEQ STOP'TEST              \ yes
560 DEX                        \ no, next vector
570 DEX
580 BPL LOOK                   \ if minus -> last vector reached
590:STOP'TEST
```

```
 600 STX TEMP              \ save X, if plus don't switch
 610 LDA@(BREAK)%256       \ set new BRKVEC
 620 STA BRKVEC
 630 LDA@(BREAK)/256
 640 STA BRKVEC + 1
 650 LDA SWITCH            \ initiate SWITCH
 660 ORA@#40               \ define special case
 670 STA SWITCH
 680 BNE TRY               \ try switched eprom first
 690:NEW'TRY
 700 LDX@#FF               \ clear stack
 710 TXS
 720 LDA TEMP              \ get flag
 730 BPL NO'NEXT           \ don't switch
 740 BIT SWITCH            \ special case?
 750 BVC NEXT              \ no
 760 STX SWITCH            \ yes, try all eproms
 770:NEXT
 780 INC SWITCH            \ next eprom
 790 LDA SWITCH
 800 CMP@ROMNUM            \ last eprom reached?
 810 BCC TRY               \ no, go try next one
 820:NO'NEXT
 830 LDA #100              \ get first letter of command
 840 JSR LOWER'CASE        \ is it a lower case letter?
 850 BCC GEN'ERROR         \ no, generate error 94
 860 LDX@0                 \ yes, convert to upper case
 870:REPEAT
 880 LDA #100,X            \ get character
 890 CMP@CR                \ is it return?
 900 BEQ RETRY             \ yes, retry
 910 JSR LOWER'CASE        \ is it a lower case letter?
 920 BCC SKIP              \ no, skip convertion
 930 AND@#DF               \ yes, convert
 940 STA #100,X            \ restore upper case letter
 950:SKIP
 960 INX                   \ next character
 970 BNE REPEAT            \ branch always
 980:RETRY
 990 JMP #C2D5             \ go interpret command
1000:GEN'ERROR
1010 LDA@(ORGBRK)%256      \ restore BRKVEC
1020 STA BRKVEC
1030 LDA@(ORGBRK)/256
1040 STA BRKVEC + 1
1050 LDX@10                \ prepare X to be non zero
1060 BIT TEMP              \ switching allowed?
1070 BPL NO'DIS            \ no
1080 STX LATCH             \ yes, show error in display
1090:NO'DIS
1100 JMP #106B             \ try basic expansion
1110:STAT
1120 JMP #C558
1130:TRY
1140 STA LATCH
1150:DON'T'SWITCH
1160 LDA #A001
1170 CMP@#BF               \ eprom present?
1180 BNE STAT              \ no
```

```
1190 JMP #A002                 \ yes, have a try
1200:BREAK                     \ new break handler
1210 PLA                       \ analogous to original
1220 PLA
1230 STA #00
1240 CMP@ERROR'94              \ error 94?
1250 BEQ NEW'TRY               \ indeed
1260 JMP ORGBRK + 4            \ original error message
1270:LOWER'CASE
1280 CMP@CH"{"                 \ is it greater than 'z'?
1290 BCS NO'LC                 \ yes, clear carry
1300 CMP@CH"a"                 \ is it less than 'a'?
1310 RTS                       \ no, carry set
1320:NO'LC                     \ yes, clear carry
1330 CLC; RTS
1340:TEST
1350 (-----------------------------------------------
1360  TEST tests if an eprom is present. If not, all
1370  bytes are equal to #FF, in which case a message
1380  is printed. The pair of zero page bytes #E8/#E9
1390  is used in the routine #F7D1 so these are free
1400  to use here as well.
1410 -----------------------------------------------)
1420 LDA@#A0; STA #E9
1430 LDY@#00; STY #E8
1440 LDA@#FF
1450:TEST'
1460 CMP (#E8),Y; BNE RTS
1470 INY; BNE TEST'; INC #E9; LDX #E9; CPX@#B0; BCC TEST'
1480 JSR #F7D1
1490 .ASCII\NO ROM AVAILABLE\,LF,CR,NOP
1500:RTS RTS
1510 .CODE #3FC               \ activate extended page
1520 (.action only to be taken in last pass: )
1530 .IF (?#28FF&1); .BYTE (BASE)/256
1540 .END OF ASSEMBLY
1550 RETURN
```

4. als een commando met een kleine letter begint (r.830) en het niet bekend was in een van de boxen, wordt de inputbuffer geconverteerd naar hoofdletters waarna een tweede poging ondernomen wordt (r.990).
5. als na alle boxen het commando/statement nog steeds niet is herkend, dan wordt dit in het schakelkaartdisplay zichtbaar gemaakt, echter alleen als er geschakeld mocht worden, aangezien het display gekoppeld is aan #BFFF (NB. i.p.v. een 7-segmentsdisplay heb ik een HP-display dat een zelftest uitvoert, d.w.z. alle 20 ledjes aan, als er een 10 naartoe geschreven wordt).
   Vervolgens wordt de controle weer teruggegeven aan P-charme die gaat kijken of er een PROGRAM is met dezelfde naam als het commando (r.1100).

De volgorde van commando's/statements ter interpretatie is dus:

basic ROM
F-blok
fl.point ROM
* P-charme (op #1XXX)
* schakelsoft (ROM, -)
ingeschakelde box
* box op 0,1,2,3,4,5,6,7
* op dit punt evt. converteren en opnieuw beginnen
* terug naar P-charme voor PROGRAM

De algemene filosofie is dat de schakelkaart en -soft voornamelijk in direct mode interessant zijn.

Het schrijven van programma's die statements van meerdere boxen gebruiken, moet worden afgeraden. Ten eerste kunnen problemen dan te eenvoudig optreden (dubbel geheugengebruik, of herdefinieren van pointers) en ten tweede wordt zo'n programma dan te veel afhankelijk van een bepaald systeem. Een kleine verandering in dit systeem (wil bij de Atom zo af en toe nog wel eens voorkomen) heeft meteen grote gevolgen.
Als in mijn configuratie alleen naar P-charme geschreven wordt, wordt de schakelkaart überhaupt niet gebruikt, terwijl ik in geval van een programma naar één of ander A-blok (bv. SALFAA, GAGS) P-charme kan (hoeft niet altijd) disabelen met ?#BFFF=#8X waarbij X het nummer is van het betreffende A-blok. De hierboven met sterretjes aangegeven onderdelen vervallen in dat geval waarna er een systeem overblijft dat vergelijkbaar is met een systeem dat "slechts" 1 toolkit bevat.
Het kan allemaal veel geavanceerder. Het nadeel daarvan is echter dat het te specifiek wordt en dat het dientengevolge te vaak moet worden aangepast. Ik heb zeker twee keer grote veranderingen moeten doorvoeren: de eerste keer toen de 80-kolommenkaart werd geinstalleerd en de tweede keer toen P-charme kwam.
Ik probeer met deze configuratie de veranderingen te beperken.
De nog te plaatsen uitbreidingen komen als "normale" A-blok toolkits op de schakelkaart en zijn dan alleen aanwezig als dat van hun verlangd wordt.

noot:
De schakelsoft zoals die hier gepresenteerd is, werkt ALLEEN met P-charme op #1XXX in combinatiemet de clubschakelkaart. Dit neemt echter niet weg dat U via aanpassingen aan en/of ideeen uit dit programma eventueel iets dergelijks voor uw eigen configuratie zou kunnen maken.

In 't formiweldig dikke Acorntjesbrood van de vorige keer stond een zoveelste symbolische assembler beschreven (SALFAA 0.4, weet u nog?). Ik vermeldde daarbij dat een volgende versie van SALFAA in SALFAA zelf geschreven zou worden. Welnu, dit was reeds een feit, nog voor het betreffende Acorntjesbrood was uitgekomen. Die nieuwe versie heette SALFAA 1.4 en inmiddels is SALFAA 1.5 zelfs beschikbaar. Om u nu niet al te veel in verwarring te brengen geef ik even een overzicht:

- versie 0.4 was (en is) bedoeld als tussenstapje om een "echte" SALFAA in zichzelf (met al z'n specifieke mogelijkheden te schrijven
- versie 1.5 is dientengevolge de versie die het beste gebruikt kan worden
- versie 1.5 is volledig "upwards compatible" met versie 0.4, zodat u al die programma's die u reeds voor 0.4 heeft geschreven met weinig of zonder aanpassingen op 1.5 kunt RUNnen

Intermezzo. Als u een beetje woordspelig bent, zou u deze nieuwe versie i.p.v. één punt vijf ook SALFAA anderhalf kunnen noemen. Dit heeft m.i. een ietwat negatieve klank, weshalve ik zo vrij ben geweest u deze kans bij voorbaat te ontnemen. Ozzemretni.

Dan nu de verschillen tussen beide versies. Deze worden niet in volgorde van belangrijkheid gegeven, doch in volgorde van willekeur.

1. De X- en Y-indices kunnen nu ook in kleine letters:

```
LDA adres,x
sta (adres),y
ldy zp,x
```

   opm. de accu blijft gerepresenteerd door de hoofdletter A; als ik ook daar een kleine letter zou toestaan zou een label nooit met een a mogen beginnen.

2. Het .GOTO directive is geimplementeerd. Dit mag gevolgd worden door een basic-label of regelnummer. Als u wilt hernummeren zult u GOTO voluit moeten typen. Voorbeeld:

```
.......
 40:DOS = FALSE   (of TRUE)
.......
100 .IF NOT(DOS); .GOTO 200
110 (DOS-routines)
.......
200 (rest van prgm)
```

   opm. behoudens uitzonderingen is een spatie achter GOTO essentieel!

3. De LIST-file is enigzins aangepast. Er wordt steeds 1 instructie afgedrukt i.p.v. de hele instructieregel.

4. Het .BYTE directive mag ook strings bevatten:

```
:ACK = 6;:CR = 13;:LF = 10
.byte ACK,"""BYTE""-string",CR,LF,"!!"
```

   de volgende bytes worden gepoked:

```
06,22,42,59,54,45,22,2D,73,74,72,69,6E,67,0D,0A,21,21
```

5. In de immediate adresseringsmode kunnen 8 bits binaire getallen worden ingevoerd. Dat zo'n getal binair moet worden geinterpreteerd wordt aangegeven door de dubbele punt (:):
LDA@:11100 is gelijkwaardig met LDA@#1C
opm. spaties zijn optioneel, dus
CPX@: 10 11 11 11 (#BF) is toegestaan.

6. Bij uitzondering mag achter .OPTION eveneens een binair getal geplaatst worden:
.OPTION :00010000;\output op smal scherm

7. De breakhandler van SALFAA blijft ook na .END actief. Deze technische uitspraak betekent dat de symbolen ook in basic te gebruiken zijn.

8. Commentaar werd in 0.4 überhaupt niet afgedrukt. In versie 1.5 wordt het commentaar achter de back slash (\) wel afgedrukt (overigens alleen in assemblermode)

9. Het afdrukken van de symbol table kan, comme il faut, gestopt worden met de shifttoets en doorgestart met de control toets. Onderbreken met de escapetoets.

10. Het .X directive. Dit is geen verschil met versie 0.4, maar ben ik vergeten om vorige keer te vertellen. Vandaar dat ik 'm hier maar even noem.
Met .TABLE worden tijdens pass 0 de eerste twee bytes van de symbol table (die het aantal aanwezige symbolen bevatten) gereset, of gecleared zo u wilt. .X (eXtern) doet hetzelfde als .TABLE, maar reset deze twee bytes niet, zodat een bestaande symbol table gebruikt kan worden. Stel dat ik reeds een tabel heb (b.v. ingeladen) op #8271, dan kan ik met .X #8271 deze "aktiveren". De nieuw te definieren symbolen worden in deze tabel bijgeschreven.

11. Bij uitzondering mag een symbool beginnen met de apostrophe ('). Dit is gedaan om problemen met basic-functies te omzeilen.
Als declaratie is dit correct:
:VALUE = #1234
Ik kan de waarde van VALUE echter niet uitlezen, omdat Atombasic de functie VAL gaat interpreteren met UE als argument.
Oplossingen:
   1. andere naam :WAARDE = #1234
   2. kleine letters :value = #1234
   3. apostrophe :'VALUE = #1234
Bedenk wel dat in PRINT- en INPUT-statements de apostrophe een betekenis heeft.
PRINT 'VALUE drukt eerst een CRLF en daarna de waarde van de VALue van UE af.
PRINT ('VALUE) drukt de waarde van 'VALUE af.
Even een opmerking tussendoor. Het gebruik van symbolen die beginnen met een hoofdletter gevolgd door kleine letters is niet mogelijk. Dit ligt niet aan SALFAA, maar aan die verrekte basic waar we mee opgescheept zitten. Dus niet :Waarde
De oplossingen zijn dezelfde als boven:
   1. andere naam :WAARDE
   2. kleine letters :waarde
   3. single quote :'Waarde

Zo draait SALFAA de LISTfile uit

```
  80 9000 40               .OPTION #40
 100 9000 03FC             .CODE #3FC              \ extension byte
 110 03FC FF               .BYTE #FF               \ don't extend
 120 03FD 2400            :BASE = #2400            \ or any other RAM page
 140 03FD 880A            :SWITCH   = #880A        \ or any other RAM byte
 150 03FD 0202            :BRKVEC   = #202
 160 03FD C908            :ORGBRK   = #C908        \ original break handler
 170 03FD BFFF            :LATCH    = #BFFF
 180 03FD 004F            :TEMP     = #4F          \ switch/no switch flag
 190 03FD 0008            :ROMNUM   =  8           \ number of eproms
 200 03FD 005E            :ERROR'94 =  94
 210 03FD 000A            :LF       = #0A
 220 03FD 000D            :CR       = #0D
 230 03FD 00EA            :NOP      = #EA
 250 03FD 2400             .CODE  BASE
 260 2400 FF E3 C6         .BYTE #FF,#E3,#C6
 270 2403 52 4F 4D         ASCII'ROM'
 280 2406 A4 0E            .DBYTE   ROM:#8000
 290 2408 20               .ASCII'-'
 300 2409 A4 23            .DBYTE DIRMO:#8000
 310 240B A4 29            .DBYTE ENTRY:#8000
 320 240D 80               .BYTE #80               \ end of 'table'
 330 240E 240E            :ROM
 340 240E 20 E1 C4          JSR #C4E1; DEC #04; LDX #04; LDA #16,X
 340 2411 C6 04             DEC #04; LDX #04; LDA #16,X
 340 2413 A6 04             LDX #04; LDA #16,X
 340 2415 B5 16             LDA #16,X
 350 2417 8D 0A 88          STA SWITCH; STA LATCH; JSR TEST; JMP #C55B
 350 241A 8D FF BF          STA LATCH; JSR TEST; JMP #C55B
 350 241D 20 CD 24          JSR TEST; JMP #C55B
 350 2420 4C 5B C5          JMP #C55B
 360 2423 2423            :DIRMO
 370 2423 20 2E 14          JSR #142E;JMP #C31B
 370 2426 4C 1B C3          JMP #C31B
 380 2429 2429            :ENTRY
 390 2429 AD 03 02          LDA BRKVEC + 1
 450 242C C9 24             CMP#(BREAK)/256        \ try to switch if BRKVEC
 460 242E F0 04             BEQ TRY'SWITCH         \  has been set to BREAK
 470 2430 C9 C9             CMP#(ORGBRK)/256       \ has BRKVEC been re-set?
 480 2432 D0 7B             BNE DON'T'SWITCH       \ yes, don't switch
 490 2434 2434            :TRY'SWITCH
 500 2434 A2 08             LDX#108                \ check if any vector
  :      :                   :                        :
1450 2407 2407            :TEST'
1460 2407 D1 E8             CMP(#E8),Y;BNE RTS
1460 2409 D0 21             BNE RTS
1470 240B C8                INY;BNE TEST';INC #E9;LDX #E9;CPX##80;BCC TEST'
1470 240C D0 F9             BNE TEST';INC #E9;LDX #E9;CPX##80;BCC TEST'
1470 240E E6 E9             INC #E9;LDX #E9;CPX##80;BCC TEST'
1470 2450 A6 E9             LDX #E9;CPX##80;BCC TEST'
1470 2452 E0 80             CPX##80,BCC TEST'
1470 2454 90 F1             BCC TEST'
1480 2456 20 D1 F7          JSR #F7D1
1490 2459 4E 4F 20 52 4F 4D 20 41 56 41 49 4C 41 42 4C 45 0A 0D EA
                           .ASCII\NO ROM AVAILABLE\,LF,CR,NOP
1500 24FC 24FE            :RTS RTS
1500 24FE 60                RTS
1510 24F5 03FC             .CODE  #3FC             \ activate extended page
1520 03FC 24               .BYTE  (BASE)/256
```

```
10REMgagsdemo
20REMv. Peter Ehrlich
30BASE#34;CLEAR4
40CREATE/P:1 0,#C1,#5D,#55,#55,#5D,#C1,0
50MOVE0,60;DRAW255,60
60BLOCK1,70,90,115,70
70MOVE60,90;DRAW195,90
80MOVE127,192;DRAW127,180
90DRAW150,165;DRAW105,165;DRAW127,180
100PAINT127,166;PAINT127,170
110FOR H=-150 TO 450 S.25
120VLINE0,H,0,127,110,60,0,A,B
130N.
140REMdivan
150MOVE78,87;DRAW178,87;DRAW185,70
160DRAW70,70;DRAW78,87
170BLOCK0,80,71,95,14
180MOVE70,70;DRAW53,57;DRAW203,57;DRAW185,70
190MOVE59,60;PLOT7,198,60
200MOVE58,59;PLOT7,199,59
210BLOCK0,53,40,151,16;CUBE1,0,53,40,151,16
220MOVE70,70;DRAW185,70
230REM kast
240BLOCK1,0,50,38,128;BLOCK0,8,50,4,50
250MOVE38,50;DRAW55,60;DRAW55,165;DRAW38,178
260DRAW38,50;MOVE40,54;PLOT7,40,174
270PAINT45,60
280MOVE0,100;PLOT7,38,100
290MOVE0,128;PLOT7,38,128
300MOVE0,150;PLOT7,38,150
310X=28;Y=110;GOS.b;X=8;Y=110;GOS.b
320X=28;Y=138;GOS.b;X=8;Y=138;GOS.b
330X=28;Y=160;GOS.b;X=8;Y=160;GOS.b
340MOVE39,60;PLOT7,53,60
350REM lamp
360MOVE225,48;DRAW235,48
370MOVE217,49;DRAW243,49
380MOVE215,50;DRAW245,50
390MOVE217,51;DRAW243,51
400MOVE225,52;DRAW235,52
410MOVE229,50;DRAW229,110
420MOVE230,110;DRAW230,50
430MOVE231,50;DRAW231,110
440DRAW250,110;DRAW242,135;DRAW218,135
450DRAW210,110;DRAW230,110
460BLOCK0,220,111,20,22;PAINT230,120
470PAINT60,99,1;PAINT200,99,1
480PAINT120,160,1;PAINT100,190,1
490PAINT60,80,1;PAINT200,80,1
500PAINT250,80,1;PAINT250,115,1
510FORN=49 TO 174 S.5
520HLINE1,39,N,130,99.200,1,X,Y
530N.
540REMbekleding divan
550FOR H=55 TO 203S.3
560 VLINE0,H,40,H,99,57,0,A,B
570 VLINE0,H,57,127,120,70,0,A,B
580N.
590FOR H=70 TO 185 S.3
600 VLINE0,H,70,127,180,87,0,A,B
610N.
620INV
630REM wolk
640X=85;Y=145;R=5;GOS.c
650X=93;Y=149;R=5;GOS.c
660X=102;Y=152;R=6;GOS.c
670X=111;Y=152;R=5;GOS.c
680X=118;Y=148;R=5;GOS.c
690X=95;Y=143;R=5;GOS.c
700X=105;Y=143;R=7;GOS.c
710X=115;Y=143;R=5;GOS.c
720DEF PLANT,0000001100110000
730DEF:       1110011000111000
740DEF:       0111011010111011
750DEF:       0011011100110111
760DEF:       1011111001101110
770DEF:       0111111001111000
780DEF:       1111111111111110
790DEF:       0111111111111100
800FOR H=74 TO 170 S.21
810 SET:PLANT,H,98;N.
820REM schip
830MOVE70,109;DRAW185,109
840MOVE90,105;DRAW165,105
850MOVE105,100;DRAW150,100
860SOUND255,255
870WINDOW1,70,90,114,70
880FOR H=50 TO 300
890MOVEH,118;PLOT6,(H-8),110
900MOVE(H-45),119;PLOT6,(H-45),123
910MOVE(H-53),124;PLOT6,(H-53),126
920MOVE(H-62),118;PLOT6,(H-60),110
930MOVE(H-58),123;PLOT6,(H-58),119
940MOVE(H-56),126;PLOT6,(H-56),124
950N.
960INV;G.860
970bREM borden
980BLOCK0,(X-2),(Y-2),5,5
990FOR R=3 TO 9
1000 CIRCLE0,X,Y,R
1010N.;R.
1020cREMschijf
1030BLOCK1,(X-2),(Y-2),5,5
1040FORN=3 TO R
1050CIRCLE1,X,Y,N
1060N.;R.
```

GAGS DEMO

GAGS DEMO

BOOTSTRAP ROUTINE

Deze bootstrap opstart routine is geschreven voor de 6502 bootstrap van Acorn Nieuws 3.7.
De routine kan los gebruikt worden maar kan ook een plaatsje vinden in de schakelsoft of in een Utilityrom. (voor mensen zonder geheugenkaart of schakelkaart). De atom zal bij gebruik van deze routine opstarten met de tekst "ACORN ATOM DOS BASIC" waneer een disc-drive aanwezig is, de disk wordt uiteraard uiteraard automatisch aangezet. Het test-byte dat hiervoor gebruikt wordt is de data #00 op adres #EFAA, dit hoeft niet altijd overeen te komen omdat er meerdere versie's bestaan. De tekst "BOOTSTRAP" die hieronder volgt kan iedereen zelf veranderen en is hier alleen als voorbeeld gekozen.
De bootstrap zal verder na een break niet meer terug springen naar de tekst-page #29 maar blijven staan waar hij stond. De oude break kan men weer verkrijgen door het samen intoetsen van shift en break.
Was er in de textpage voor de break nog een programma aanwezig dan dan zal er na de break informatie over de textpage en de top van het programma verschijnen. Men kan nu zonder old of end direkt weer verder werken aan datzelfde programma.
De plaats van de routine in het geheugen is niet belangrijk echter deze uitvoering is zodanig geschreven dat hij direkt werkt bij een beginadres van bijv. #1F00, #2F00, #3F00 enz. Wilt men de routine toch op een andere plaats zetten dan zullen de interupt-vectoren niet direkt achter het programma gezet kunnen worden omdat deze niet de laatste 6 bytes van een 4Kbyte geheugen blok zijn. Men zal nu zelf de vectoren in het geheugen moeten zetten. Voor de instelling van de bootstrap wordt verwezen naar het artikel "bootstrap" in de Acorn Nieuws van december 1984.
Verder nog veel plezier met deze routine:

```
   10REM BOOTSTRAP ROUTINE
   20REM VERSIE 2.5
   30REM CHARL DE MOOR
   40REM DATE: 18.12.84
   50 DIMLL9,A8,B8;@=0;F.L=0TO9;LLL=#FFF;N.
   60 P.$6$12"BOOTSTRAP ROUTINE"'"VERSIE 2.5 CHARL DE MOOR"''
   70 P." VOORBEELD STARTADRES: #.EED"'" '.'= PAGINA: 0-F"'
   80 P." BIJ ANDER START ADRES INTERUPT"'" POINTERS ZELF"
   90 P."AANPASSEN"'" ZIE A.N. DEC.1984 BLZ.68"''
  100 IN."START ADRES :"Z;IF?18=Z/256;G.100
  110 IN."PRINTER (J/N)"$A
  120 IN."LISTING (J/N)"$B
  130 P.'"PASS:1"$21;F.I=1TO2;P=Z;?#E8=0;IFI=1G.170
  140 P.$6$8I''$21
  150 IF?A=CH"J"P.$2
  160 IF?B=CH"J"P.$6
  170[
  180:LL0;LDX@#17;LDA#FE;CMP@#0A;BEQ LL1
  185 LDA@#0A;STA#FE;LDA@#29;STA#12
  190:LL1;LDA#FF9A,X;STA#204,X;DEX;BPLLL1;TXS;TXA;INX
  200 STX#EA;STX#E1;STX#E7;STX#CB;LDX@#33
  210:LL2;STA#2EB,X;DEX;BPLLL2;LDA@#07;STA#B002;LDA@#8A;STA#B003
  220 JSR#F7D1;];!P=#0F0C06;P=P+3
  230 $P=" ACORN ATOM ";P=P+LENP;[;LDA#EFAA;BNELL3
  240 JSR#E000;JSR#F7D1;];$P="DOS ";P=P+LENP;[;NOP
  250:LL3;JSR#F7D1;];$P="BASIC";P=P+LENP;!P=#0D0A;P=P+2
  260 $P=" BOOTSTRAP";P=P+LENP;!P=#0D0A0A;P=P+3;[
  270 LDA#B001;CMP@#7F;BNELL4 SHIFT TOETS ?
  280 JSRLL8;JMP#C2B2
```

```
290:LL4;LDA#12;STA#0E;LDA@#01
300 LSRA;STA#0D;TAY;LDA(#0D),Y;CMP@#0D
310 BNELL5;INY;LDA(#0D),Y;BPLLL6
320:LL5LDY@#00;LDA@#0D;STA(#0D),Y;INY;LDA@#FF;STA(#0D),Y;INY
330 STY#0D;JSRLL8;JMP#C2B6
340:LL6;LDY@#00
350:LL7;INY;BEQLL5;;LDA(#0D),Y;CMP@#0D;BNELL7
360 JSR#CDBC;LDA(#0D),Y;BMILL9;INY;BNELL7
370:LL9;JSR#F7D1;];!P=#0D0A;P=P+2;$P=" TEXT POINTER: #"
380P=P+LENP;[;NOP;LDA#12;JSR#F802;JSR#F7D1;];!P=#0D0A;P=P+2
390 $P=" TOP OF       : #";P=P+LENP;[
400 LDA#0E;JSR#F802;LDA#0D;JSR#F802;JSR#FFED;JSRLL8;JMP#C2CA
410:LL8;JSR#F7D1;];$P=" READY";P=P+LENP
420 !P=#0D0A0A;P=P+3;[;NOP;RTS
430];N.;IF?#E8=#23;G.470
440 !P=#FFC70206;P=P+4;?P=Z%256;P?1=Z/256;P=P+2;!P=#FFB2;P=P+2
450 IFP%1024<>0P.$6"INTERUPT POINTERS AANPASSEN !"
460 P.$6'"READY"''"TOP OP: #"&P'"LENGTE: #"P-Z"  BYTES"'';END
470 P.$6$7$7$7'"OUT OF RANGE DETECTED"'';END
```

LET OP:
Het kan zijn dat de routine bij gebruik met sommige schakelsoften problemen oplevert.
Mochten er onverhoopt problemen zijn of heeft U nog suggesties, bel of schrijf dan even naar C. de Moor of E. Sanders.

TIP 1

Voor het gebruik van de door M. Janssen in Acornnieuws 3e jaargang no. 5 beschreven verbeterde 1200 baud leesroutine moeten een aantal door hem aangegeven wijzigingen in P-Charme aangebracht worden.

Als je nu niet direct een Eprom-programmer in de buurt hebt (zoals ik), dan kun je toch deze nieuwe leesroutine gebruiken door de oude routine naar een lees stukje RAM te copieren, daar de wijzigingen aan te brengen en tot slot de atom te vertellen waar de nieuwe leesroutine staat.

Dit gaat heel gemakkelijk met het volgende programma'tje:

```
10 PROGRAM FCOS
20 COPY #ADE0,#AF34,#WXYZ
30 P=#WXYZ+#13
40 !P=#BD200818;P!4=#FCBF20FC;P!8=#06B012E0
50 P!12=#3804B028;P!16=#EA28EFB0
60 COS 1
70 ?#214=#YZ;?#215=#WX
80 END
```

MET #WXYZ BIJV. #6000(ZODAT YZ=00 EN WX=60)
Na een break even opnieuw runnen.

INPUTSUBROUTINES

Onderstaande subroutines zijn bruikbaar in alle BASIC programma's. Zij dienen ervoor om een gebruikersvriendelijke input van getallen of strings te bewerkstelligen. De eerste subroutine is een voorbeeldprogramma om getallen in te voeren. Door middel van een matrix worden deze getallen ingevoerd. De dimensies van de matrix worden bepaald door een aantal variabelen in regel 190 en 260:

H (htab) : aantal plaatsen vanaf links
V (vtab) : aantal plaatsen vanaf boven
K (kolom) : aantal kolommen in de x-richting
R (regel) : aantal regels in de y-richting
B (breedte): de kolombreedte

De grootte van het in te voeren getal is afhankelijk van de kolombreedte. Met de toetsen A, Z, [en] kan de cursor verplaatst worden naar ieder element van de matrix. Als op een bepaalde plaats een getal ingetikt wordt, kan er door middel van de cursorbediening het getal in de tweedimensionele array %II gevoerd worden. Een fout ingevoerd getal wordt eenvoudig verbeterd door met de cursor naar het getal te gaan en een nieuwe waarde in te tikken. Als men de invoersubroutine wil verlaten kan men op toets E drukken. De gegevens in de array %II kunnen dan verwerkt of opgeslagen worden.

De tweede subroutine is er een om strings, die reeds bekend zijn in het programma, aan te roepen zonder daarvoor de gehele string te hoeven intikken. Als er bijvoorbeeld om een naam gevraagd wordt, zal de computer bij het intikken van de 'H' direkt reageren door de naam 'HANS' te printen als deze naam in de stringarray voor zou komen. Is er echter nog een naam die begint met een 'H' dan zal de computer wachten tot er een tweede of derde toets wordt ingedrukt, net zolang totdat deze een string heeft gevonden, die verschillend is dan de anderen. Bedenk wel dat als deze subroutine gebruikt wordt voor een stringarray van 100 elementen dat het wel eens erg lang kan duren voordat de computer alles gekontroleerd heeft. Dit is helaas te wijten aan de slechte stringfunkties van de Atom. Met een LEFT$ zou deze subroutine veel sneller zijn.

```
 10 PROGRAM INPUTSUBROUTINE 1
 20 REM Frans le Blanc
 30 DIM A(10),LL(10)
 40 FDIM %II(4,9)
 50 FOR N=0 TO 10;LLN=-1;NEXT
 60 GOTO 260
 70
 80 REM voorbeeldmatrix: 5*10
 90 PRINT $12
100 HTAB10;I=65;@=5
110 F.N=0TO4;P.$(I+N)"    ";N.;P.'
120 FOR N=0 TO 9;PRINT " "N
130   FOR J=0 TO 4
140     %II(J,N)=0;PRINT %II(J,N)
150   NEXT;PRINT '
160 NEXT
170 P.'"CURSORBESTURING:"'
180 P."A - OMHOOG     B - OMLAAG"'"[ - LINKS      ] - RECHTS"
190 INVOER(10,2,5,10,5)
200 P.$12;@=5
210 F.N=0 TO 9
```

```
220   PRINT " "N;F. J=0 TO 4;P. %II(J,N);N.;P.'
230 NEXT
240 END
250
260 PROC INVOER(H,V,K,R,B)
270 X=H;Y=V;CU.X,Y
280 C=-1;I=0;$A=""
290 DO C=C+1
300   INKEY I
310   %IF I<>65 A. I<>90 A. I<>91 A. I<>93 A. I<>69
320     %IF C<B-1
330       %IF I=CH"." OR I=CH"-" OR (I>=CH"0" AND I<=CH"9")
340         GETAL;CU.X,Y
350       ELSE BEEP 20,30;C=C-1
360     ELSE BEEP 20,30;C=C-1
370   ELSE TABCURSOR
380 UNTIL I=CH"E"
390 PEND
400
410 PROC TABCURSOR
420 IF C>0;%II(((X-H)/B),(Y-V))=VAL(A);$A=""
430 C=-1
440 IF I=CH"[";X=X-B
450 IF I=CH"]";X=X+B
460 IF I=CH"A";Y=Y-1
470 IF I=CH"Z";Y=Y+1
480 IF X<H;X=(K-1)*B+H;Y=Y-1
490 IF X>(K-1)*B+H;X=H;Y=Y+1
500 IF Y<V;Y=R+V-1;X=X-B;GOTO 480
510 IF Y>R+V-1;Y=V;X=X+B;GOTO 480
520 CU.X,Y
530 PEND
540
550 PROC GETAL ,X,Y
560 X=X-B+1;CU.X,Y
570 A?C=I;A?(C+1)=#0D
580 F.L=1 TO B-LEN(A);P." ";N.
590 PRINT $A
600 FEND
610
620 GOTO 90
```

```
 10 PROGRAM INPUTSUBROUTINE 2
 20 T=10
 30 DIM VV(T),UU(T)
 40 F.N=0 TO 10;DIM U(10);UU(N)=U;N.
 50 GOTO 180
 60
 70 REM voorbeeld 10 strings
 80 P.$12"            VOORBEELD"''
 90 F.N=1 TO T;READ $UU(N);N.
100 DO
110 P."TYP EEN STRING IN: "
120 INVOER;P.'
130 U.0
140 END
150
160 om deze subroutine gaat het !!
170 ***********************************************************
```

```
180 PROC INVOER
190 K=0
200 F.N=1 TO T;VVN=0;N.
210 INKEY I;IF I>#60;I=I-#20
220 IF I<CH"A";BEEP 20,30;G.210
230 J=0
240 F.N=1 TO T
250 IF VVN=0;IF I=UUN?K OR I=UUN?K+#20;J=J+1
260 N.
270 IF J=0;BEEP 20,30;G.210
280 J=0;P.$I
290 F.N=1 TO T
300  XIF VVN=0
310   XIF I=UUN?K OR I=UUN?K+#20;J=J+1;M=N
320   ELSE VVN=1
330  ELSE
340 N.
350 IF J>1;K=K+1;G.210
360 F.N=0 TO K;P.$8;N.
370 P.$UUM
380 PEND
390 ***********************************************************
400
410 GOTO 70
420
430 DATA "PIET","KLAAS","JAN","ATOM"
440 DATA "KOOS","PETER","CORNELIS"
450 DATA "MARJAN","MARIA","ACORN"
460 variabelen:
470
480 ARRAY :       VV
490 STRING:       UU
500 AANTAL:       T
510 DUMMY :       I,J,K,M,N
```

CURSOR x,y

Met het statement CU(RSOR) x,y kan de cursor op iedere willekeurige plaats op het beeldscherm gezet worden. Ook in tekstmodes met meerdere karakters zoals de 40 * 24 of 64 * 24 werkt onderstaand statement.

```
 10REM CURSOR voor alle modes
 20REM ook voor GRMOD, VDU 40*24 of 64*24 etcetera
 25REM Frans le Blanc
 30DIM LL4
 40IN."ROUTINE OP :"Q
 50F.N=0TO4;LLN=0;N.
 60P.$21
 70F.N=0TO1;P=Q;[
 80:LL0
 90      JSR#C78B;JSR#C231
100      JSR#C4E1
110      LDX@0;STX4
120      LDA@30;JSR#FFF4
130      LDA#17;TAY;BEQ LL2
140      LDA@#0A
150:LL1
160      JSR#FFF4;DEY;BNE LL1
170:LL2
180      LDA#16;TAY;BEQ LL4
190      LDA@#09
200:LL3
210      JSR#FFF4;DEY;BNE LL3
220:LL4
230      JMP#C55B
240];N.;P.$6$11;@=0
250P."ROUTINE OP : #"&Q"-#"&P'
260E.
```

Schakelkaart problemen.

Laatst belde mij iemand die problemen had met zijn nieuwe schakelkaart. Alles zou volgens het boekje aangesloten zijn. (ACORN NIEUWS '84 nr 7: blz 49). Er werkte echter niets...

Na een avond rondspitten op de nieuwe kaart kwamen we tot de volgende conclusie:
De kaart heeft een nieuwe adresdecoder gekregen n.l. een 75LS154. Dit is een 4 naar 16 lijnen decoder. Door dit IC aan te sluiten volgens fig. 1 kunnen we het gehele adresgebied van 0XXX tot FXXX uitdecoderen. Normaal ziet dit er als in fig. 1 getekend uit. Het verhaal uit ACORN NIEUWS (blz. 49) zou dan ook kloppen.

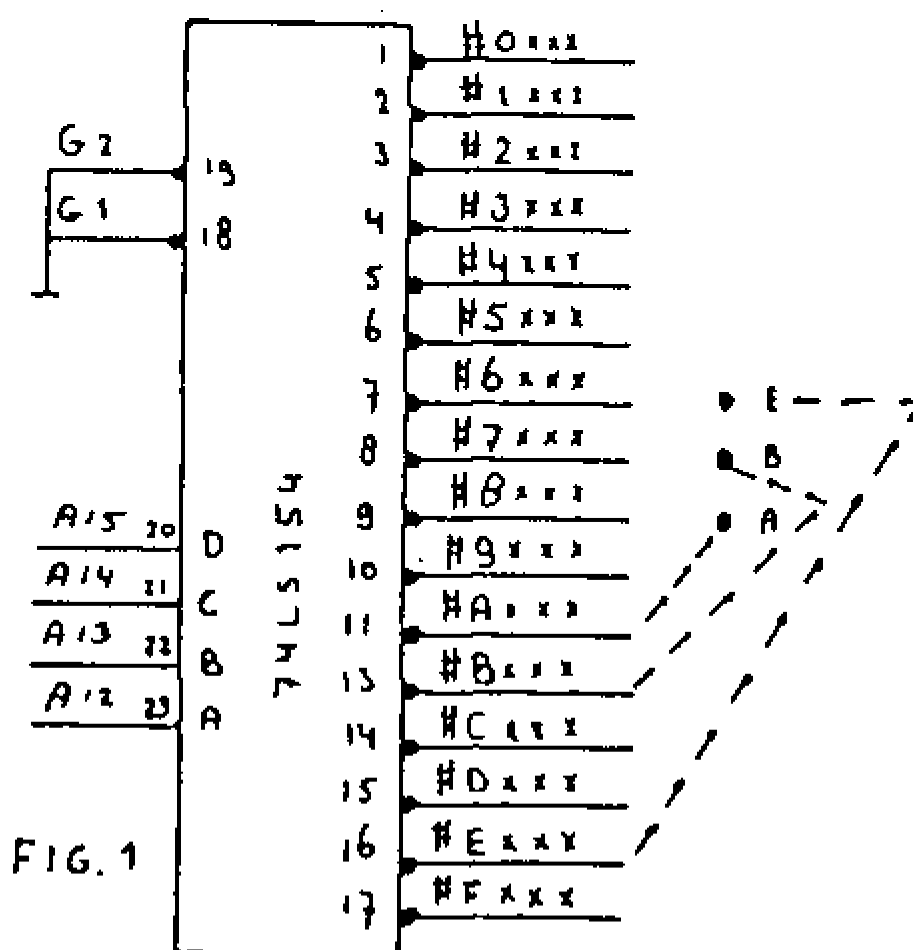


FIG.1

ZO, ZIT HET ECHTER NIET, HELAAS.

In werkelijkheid zijn de adressen A15, 14, 13 en 12 omgedraaid, waardoor de hele decodering toch wel iets anders wordt.
Dus het E blok zat op 7xxx
B blok zat op 3xxx
A blok zat op 5xxx

Volgens de werkelijke (fig. 2) zit het dus z0:
E blok naar pin 8
B blok naar pin 15
A blok naar pin 6

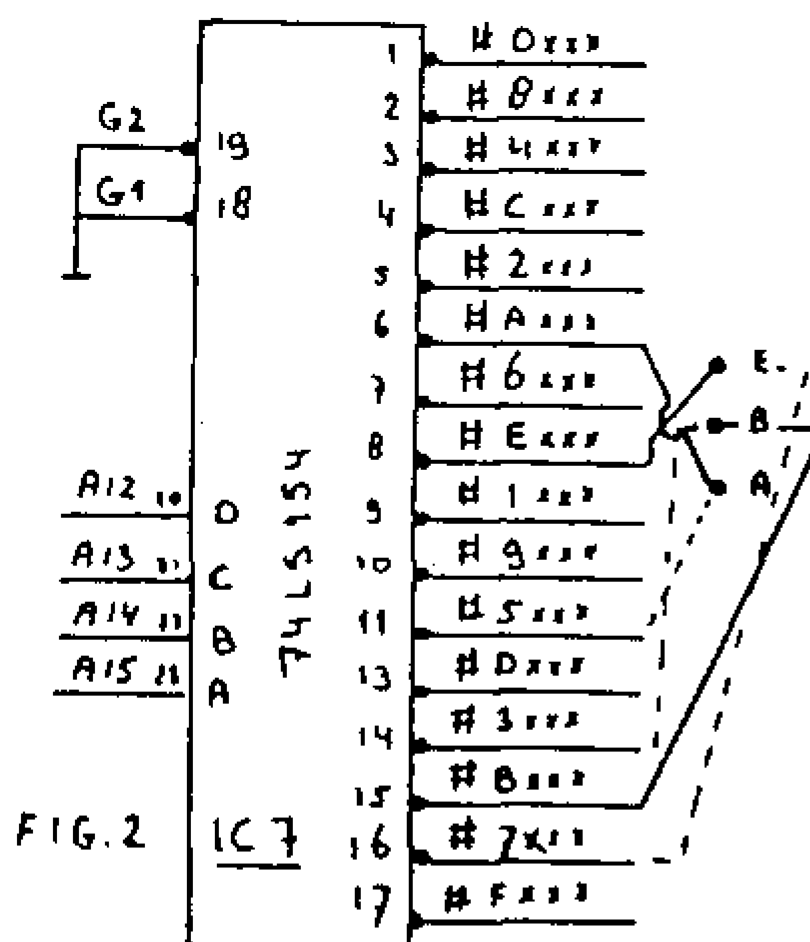


FIG.2

Toen we deze veranderingen hadden gemaakt, werkte de kaart perfect.

't is maar dat je het weet.

*** REGRESSIE ***

Op een gegeven moment bestond er bij mij de behoefte om van een aantal zaken de beste functie te bepalen. Wiskundig kan dat op een aantal manieren, bijv. door de polynoom te bepalen.
Een nadeel van deze oplossing is, behalve dat zelfs een computer er erg lang over doet om bijv. een 50-ste graadsfunctie te bepalen, dat niet altijd een vloeiende lijn verkregen wordt. Bovendien worden foutieve waarnemingen in een reeks getallen volledig meeberekent.
Een andere, veel bekendere oplossing is die m.b.v. regressie. Maar veel (natuurkundige) relaties zijn echter niet lineair: in de natuur komen erg veel logaritmische of exponentiele verbanden voor. Natuurlijk is het mogelijk om de logaritmische, exponentiele of machts-regressies en correlaties te bepalen. In het door mij geschreven programma wordt de regressielijn van een variabele Y uit de verzameling X bepaald. Gekozen kan worden voor zowel lineaire, logaritmische, expontiele als machts-regressie. Uiteraard is de lijn die bepaald wordt met de hoogste absolute correlatiecoefficient de meest waarschijnlijke funktie van Y uit X.
Het programma heeft een menu, dat voor zich spreekt.
Uiteraard is het mogelijk om ingevoerde data op schijf te zetten en later weer te laden en eventueel veranderingen in aan te brengen. Ook is het mogelijk om grafieken op het beeldscherm te tekenen. Het plotten kan onderbroken worden met de shift-toets.
Het programma is vrij lang, ca. 10K, dus een 16K-kaart is wel een vereiste. Verder moet U een P-charme hebben en een TXMOD en GRMOD (nodig voor het tekenen van grafieken). Verder kunt U statements als INVERT, CLS, FAST gewoon weglaten als U deze niet in een eventuele schakelkaart heeft. Verder is het nodig om als U GRMOD net als ik ook in eprom op #A000 heeft, om deze box steeds in te schakelen: mij bij gebeurt dat met het statement EP(ROM).
Als U over een matrixprinter beschikt kunt U de meeste controlcodes in dit programma waarschijnlijk gebruiken: deze zijn voor een AVT-printer. U kunt als U over een graphics-dump voor Uw printer beschikt deze inbouwen na regel 1600.

data X : .165
data Y : .165
DATA

	X	Y		X	Y
1	3.000000	0.988000	2	4.000000	0.900000
3	5.000000	0.805000	4	6.000000	0.729000
5	7.000000	0.669000	6	8.000000	0.622000
7	9.000000	0.582000	8	10.000000	0.550000
9	11.000000	0.521000	10	12.000000	0.497000
11	13.000000	0.476000	12	14.000000	0.458000
13	15.000000	0.441000	14	16.000000	0.426000
15	17.000000	0.412000	16	18.000000	0.400000
17	19.000000	0.389000	18	20.000000	0.378000
19	21.000000	0.369000	20	22.000000	0.360000
21	24.000000	0.344000	22	26.000000	0.330000
23	28.000000	0.317000	24	30.000000	0.306000
25	35.000000	0.275000	26	40.000000	0.264000
27	50.000000	0.235000	28	60.000000	0.214000
29	70.000000	0.195000	30	80.000000	0.185000
31	90.000000	0.173000	32	100.000000	0.165000

LINEAIR

AANTAL PAREN = 32
CORRELATIECOEFF. = -0.761518
COEFFICIENT A = -0.006164
COEFFICIENT B = 0.606831

DE REGRESSIELIJN VAN Y UIT X IS: Y = -0.006164 * X +0.606831

LOGARITME

AANTAL PAREN = 32
CORRELATIECOEFF. = -0.963639
COEFFICIENT A = -0.223584
COEFFICIENT B = 2.983316

DE REGRESSIELIJN VAN Y UIT X IS: Y = -0.223584 * ln X + ln2.983316

EXPONENT

AANTAL PAREN = 32
CORRELATIECOEFF. = -0.899411
COEFFICIENT A = -0.016527
COEFFICIENT B = -0.479911

DE REGRESSIELIJN VAN Y UIT X IS: $Y = e^{(-0.016527 * X \ -0.479911)}$

MACHT

AANTAL PAREN = 32
CORRELATIECOEFF. = -0.999673
COEFFICIENT A = -0.526503
COEFFICIENT B = 1.839570

DE REGRESSIELIJN VAN Y UIT X IS: $Y = 1.839570 * X_{-0.526503}$

STATISTIEK

AANTAL PAREN = 32
COVARIANTIE = 8.065531
GEMIDDELDE X = 27.593750
GEMIDDELDE Y = 0.436718
STAND.DEV. X = 25.831787
STAND.DEV. Y = 0.209122
MAXIMUM X = 100.000000
MAXIMUM Y = 0.988000
MINIMUM X = 3.000000
MINIMUM Y = 0.165000

Acorn Atom-computer

In Acorntjesbrood 2.2 heb ik op pag.7 en 8 het één en ander verteld over het implementeren van BBC Basic op de Atom zonder de BBC-kaart de gebruiken. Het aldaar voorgestelde plan om door het relocaten van de 20K ROM van de BBC-kaart het mogelijk te maken om in een willekeurige Atom deze 20K in te laden in het geheugen en dan BBC Basic te kunnen draaien was weliswaar nog lang niet afgerond, maar toch wilden wij U er alvast het één en ander over vertellen. (Projectgroepen, meldt de stand van zaken in Acorntjesbrood!).
Toen door het beschikbaar komen van P-Charme onze animo om nog veel tijd aan dit project te besteden al flink gedaald was (procedures, functies (beide zelfs nog krachtiger dan in BBC Basic), meerdimensionale arrays, controlstructures en nog veel meer, beschikbaar voor èlke Atom), viel als klap op de vuurpijl bij mij het aprilnummer van de Stacker (regio Noord) in de brievenbus (voor het drukwerkarchief, U weet wel). Hierin las ik met rode oortjes een beschijving van Marien van Westen hoe hij met een aantal hardware wijzigingen (kosten ongeveer 25 gulden) BBC Basic op zijn Atom draait. (Dus zonder het relocaten van de ROM's). In principe voert hij de wijzigingen zoals die op de BBC-kaart plaatsvinden in een veel eenvoudiger vorm uit.
In het licht van deze ontwikkeling viel al vrij snel het besluit te stoppen met het relocaten van de ROM's (ja inderdaad, zonde van de tijd.)

De hardware modificaties van Marien (voor geïnteresseerden: zie het drukwerkarchief) waren niet zonder meer toepasbaar in mijn Atom, omdat hij zijn #AXXX-blok binnen de bus heeft en ik dat buiten de bus heb op de schakelkaart. (Ik zal hier maar niet verder op de details ingaan).
Na enig experimenteren en solderen lukte het Bram en mij om op mijn Atom BBC Basic te draaien (!). We hebben het basis-idee van Marien gebruikt en zijn uiteindelijk op een vrij nette en eenvoudige ombouw uitgekomen.
Voor ik die uit de doeken doe zal ik even mijn systeem (voor zover relevant) beschijven:

- RAM van #2000-#7FFF, waarvan #4000-#7FFF op de geheugenkaart zit.
- Club-schakelkaart, waardoor het voetje voor IC24 op het moederboard leeg is.
- VIA (6522). Deze is nodig voor o.a. de statements SOUND en TIME.

Indien U de hierna volgende wijzigingen niet begrijpt en niet weet wat U doet, kunt U er maar beter niet aan beginnen.

Het implementeren van BBC Basic is op te splitsen in 3 afzonderlijke delen:

1) Door het in BBC Basic stand verwisselen van adreslijn A14 en A15 komt de geheugenkaart (#4000-#7FFF) op #8000-#BFFF en omgekeerd. We kunnen dan de BBC Basic interpreter inladen op #4000-#7FFF en die komt dan in BBC stand op de goede adressen (nl.#8000-#BFFF). Om te voorkomen dat door schrijffouten de interpreter wordt overschreven,is het handig de geheugen-

kaart op write-protect te kunnen zetten.
Ikzelf gebruik voor het omschakelen tussen Atom en BBC Basic een driepolige omschakelaar,waarvan 2/3 voor het verwisselen van de adreslijnen wordt gebruikt, en wel als volgt:

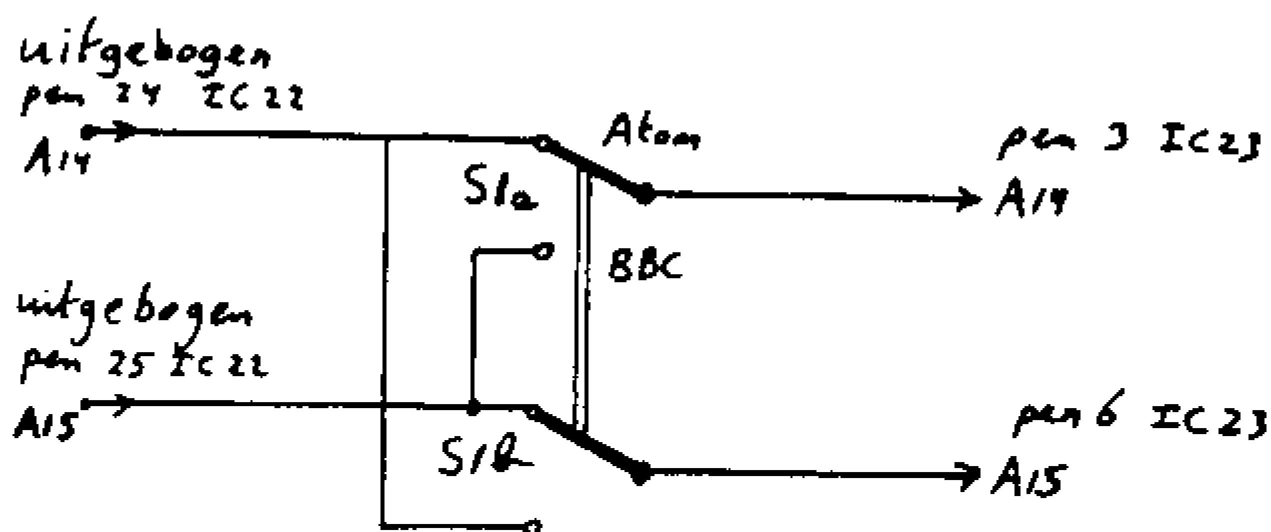


fig. 1. Verwisselen A14 en A15.

2) Het derde wisselkontakt van de schakelaar hebben we nodig om te kunnen kiezen tussen het normale Atom F-blok en de BBC-MOS-ROM, die ook op #FXXX moet zitten. Deze BBC-MOS-ROM bevindt zich bij mij in een 2732 eprom die via een tussen-voetje in de tot nu toe lege IC24-voet op het moederboard zit (leeg vanwege de schakelkaart). Hij kan natuurlijk ook in een 2532.Via een tussenvoetje dan pen 20 van de 2532 (CS) uitbuigen.
We kunnen tussen deze 2 F-blokken als volgt omschakelen:

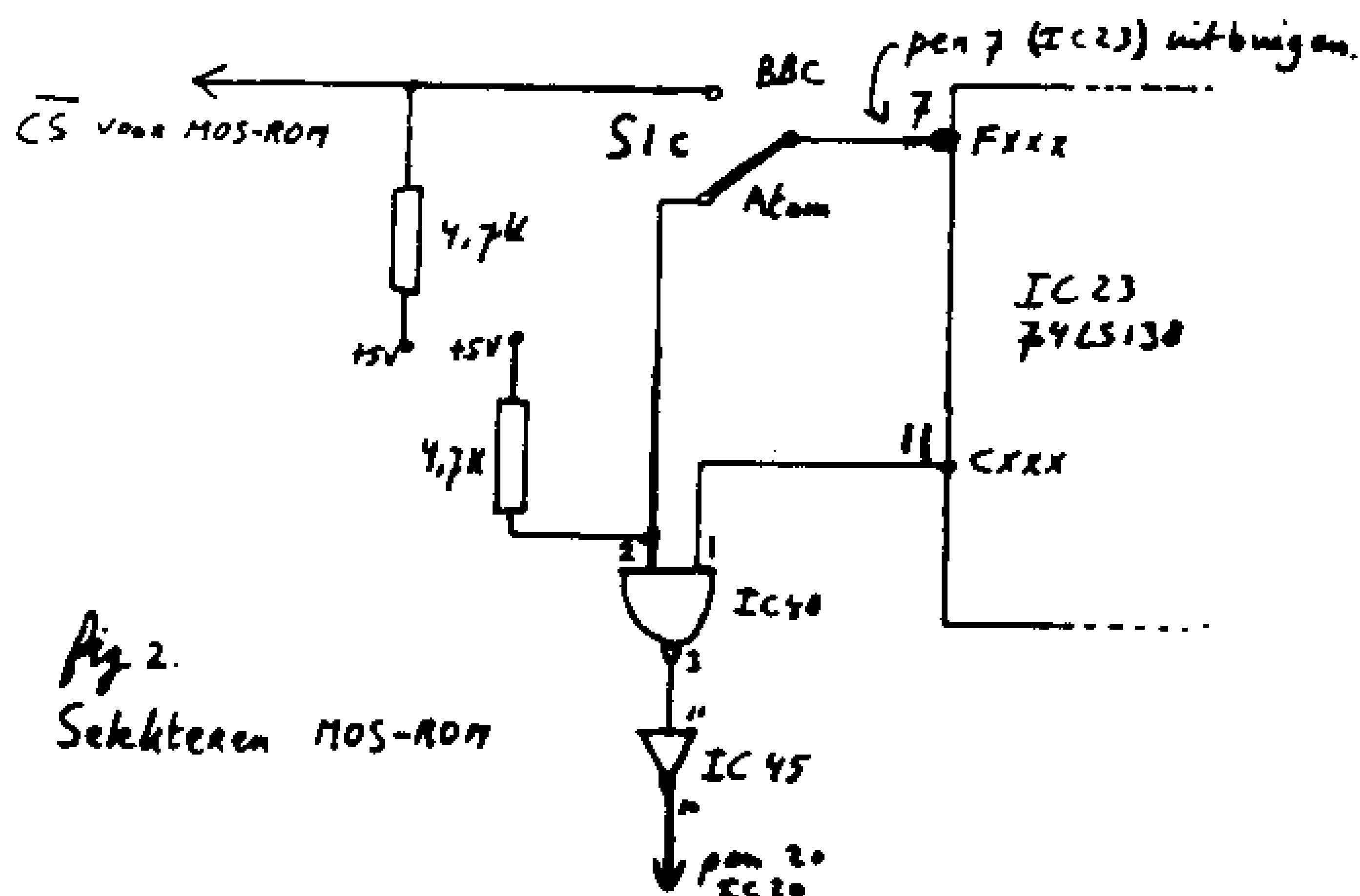


fig 2.
Sekkteren MOS-ROM

3) Verder moeten we RAM hebben op #0400-#07FF. Dit wordt als werkgebied gebruikt.
Eén van de vele manieren om dit te krijgen is de volgende:
Stapel twee 2114's in het lage tekstgeheugen. Verbind de

beide uitgebogen CS-lijnen (pen 8) met onderstaande decodeerschakeling:

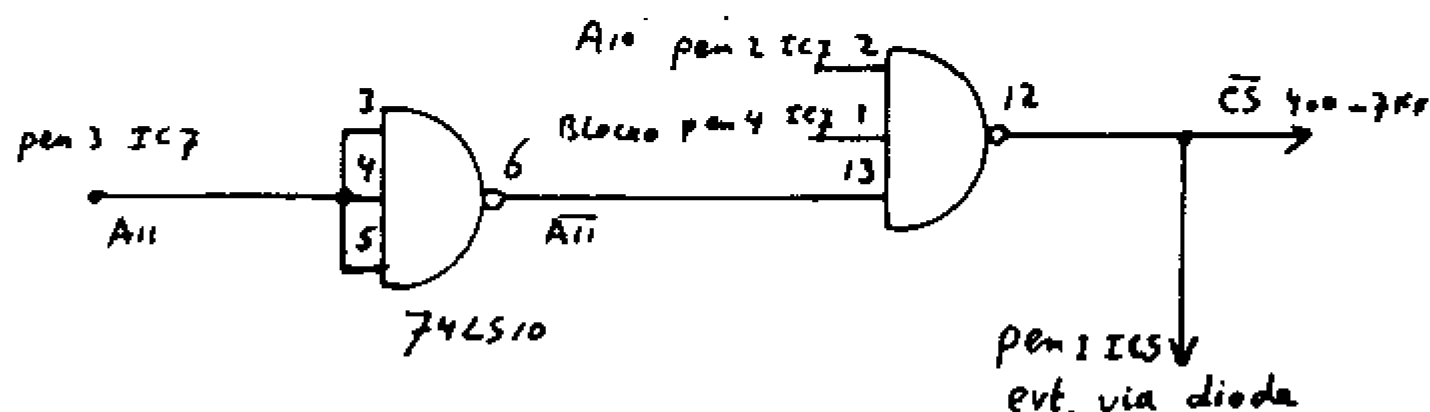


fig. 3A. Uitdecoderen #400-#7FF

Het gebruikte IC (74LS10) kan bijvoorbeeld worden gestapeld op IC7 op het moederboard of,als Uw kast dan niet meer sluit omdat de schakelkaart of de geheugenkaart in de weg zit, op bijv. IC48.
Op IC7 gaat de zaak er als volgt uitzien:

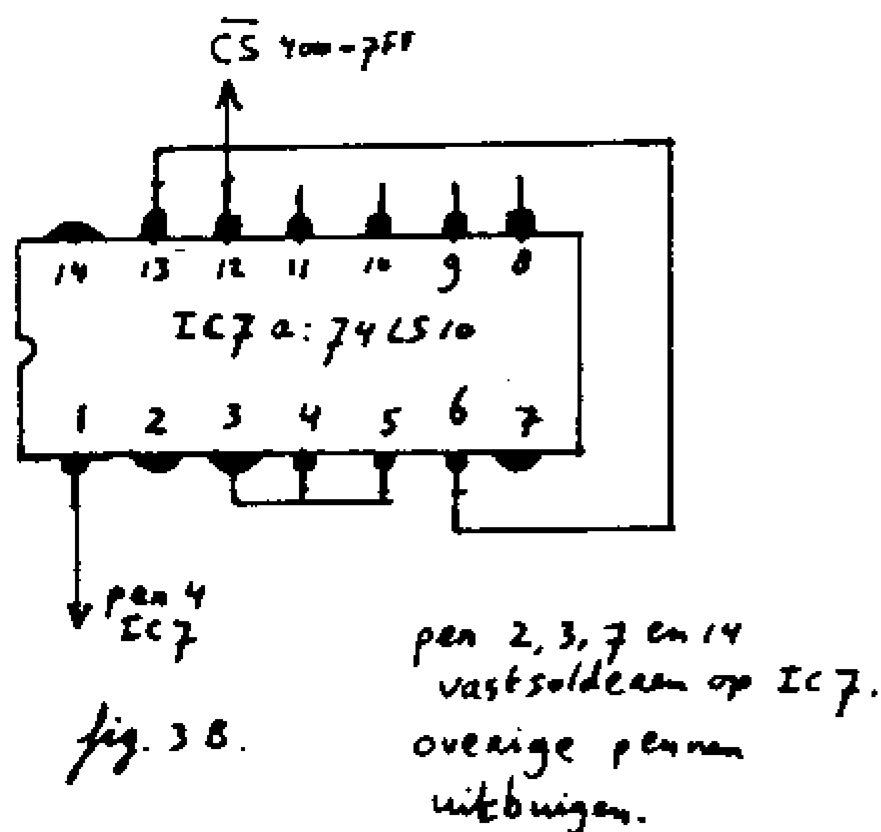


fig. 3B.

Van het geheugengebied #2000-#7FFF wordt dus #4000-#7FFF voor de BBC Basic interpreter gebruikt. Het gebied #2000-#3FFF (8K) is beschikbaar voor het BBC Basic programma. In de BBC-MOS-ROM moet op adres #F0FF en #F107 het high-byte van het begin van dit gebied worden ingevuld (bij mij dus #20). Heeft U geen gestapeld laag geheugen (#2000-#27FF) dan dus #28. Heeft U RAM op #1XXX of nog lager, dan kan dat natuurlijk ook gebruikt worden. Het BBC programma mag zelfs nog op #0800 beginnen.

Na het uitvoeren van deze wijzigingen moet de BBC Basic interpreter ingeladen worden op #4000 e.v. Terwijl we de <BREAK>-toets ingedrukt houden, zetten we de driepolige omschakelaar om, laten de <BREAK>-toets los en drukken er nog een keer op. Nu moet de tekst: BBC BASIC en de ">"-prompt op het beeldscherm verschijnen.

U kunt nu BBC Basic gebruiken. De bij de BBC Basic kaart geleverde "manual" is erg summier en slecht, zodat een extra boek over BBC Basic eigenlijk wel nodig is. Ikzelf heb het nederlandse boek "BBC Microcomputer" van A.Sickler (prijs F.29,50, uitgeverij Kluwer) aangeschaft.
Alle op BBC hardware gebaseerde statements, zoals ENVELOPE, ADVAL, COLOUR enz. kunnen we uiteraard niet gebruiken. De meeste BBC programma's uit tijdschriften (spelletjes) maken hier gebruik van en moeten worden aangepast of zijn überhaupt niet te runnen door bijv. bepaalde VDU-statements. U hoeft BBC Basic dáárvoor dan ook niet te implementeren. Wèl vanwege ondermeer lange variabelen (integer en floating point), Microsoft-achtige stringhandling, de hoge executiesnelheid en de diverse geavanceerde statements. (Beschouw bijv. eens het instellen m.b.v. @% van de wijze waarop getallen met PRINT worden afgedrukt).

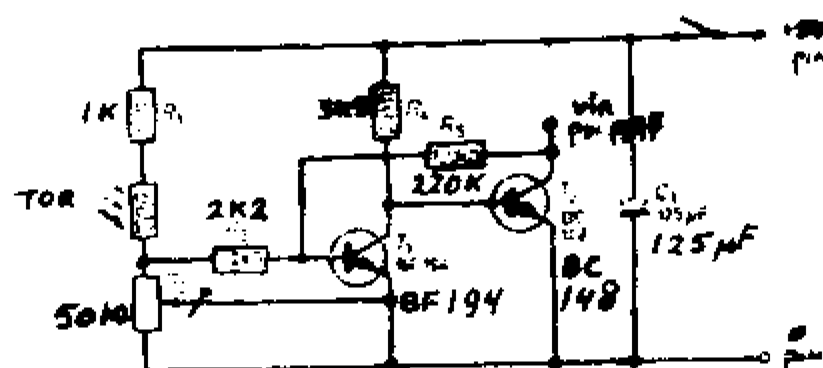
Het is niet eenvoudig te realiseren dat er in BBC-stand gebruik van de Atom-DOS kan worden gemaakt. Hiervoor moet de DOS-ROM sterk gewijzigd worden. Dus files in Atom-stand inladen of saven en dan weer terugschakelen.
Op dit moment ben ik bezig met het aanpassen van de BBC-MOS-ROM zodat de COS op 1200 baud werkt (met dank aan Marien) en een aantal andere wijzigingen. De belangrijkste hiervan is het aanbouwen van de 40*24-kolommen VDU van Paul Kuyper (AcornNieuws 2.4 pag.70) in een aan de BBC-MOS-ROM aangepaste versie. Hierdoor zijn kleine letters en een standaard scherm van 24 regels van 40 kolommen beschikbaar. Dit komt o.a. van pas bij Basicode-2, waarvoor Marien een leesroutine heeft gemaakt. (Basicode-2 op de Atom!). Hierover kunt U alles lezen in het volgende nummer van Acorntjesbrood.

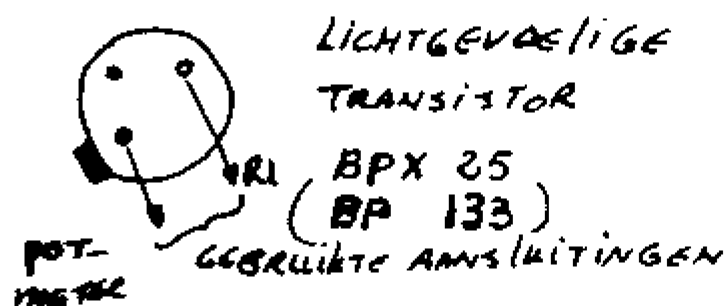
LICHTPEN AAN DE ATOM

Sinds een jaar heb ik, tot volle tevredenheid, een lichtpen aan mijn atom hangen. De pen kan verticaal 192 posities (lijnen) onderscheiden (kan meer maar lijkt me niet nodig).
In een programma met b.v. enkele menu's is het erg goed te gebruiken (tot 16 keuze's).

De hardware

Om de benodigde lichtsterkte te kunnen regelen heb ik de volgende schakeling uit een Philips EE-doos gebruikt (misbruikt).

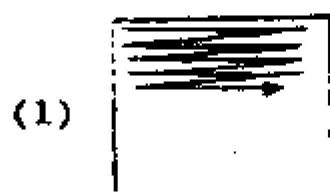


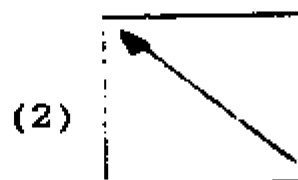


De schakeling is geplaatst aan de VIA. de lichtgevoelige transistor met ingebouwd lensje is een pen geplaatst.

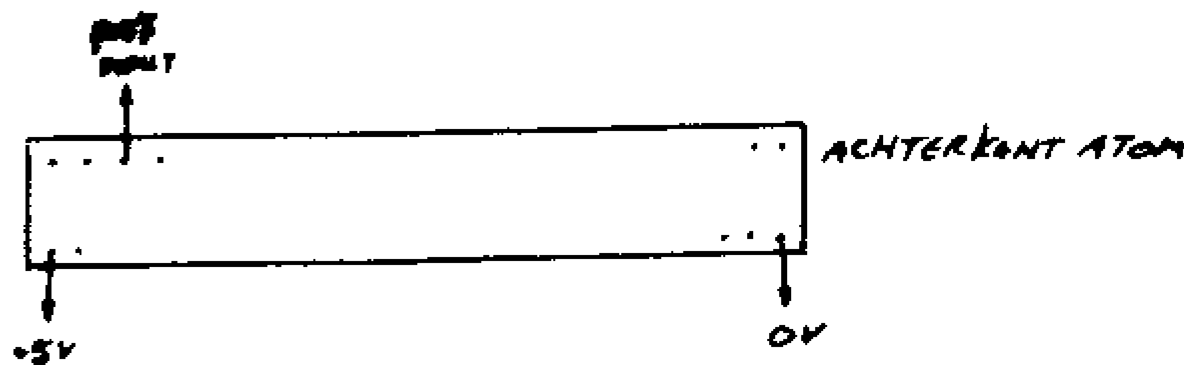
Globale werking

Adres #B800 van de VIA geeft aan of de elctronenstraal (lichtpuntje) een beeld aan het schrijven is (1) of dat deze van onder naar boven terug gaat (flyback) (2)

(1) 

(2) 

Dit schrijven gebeurt een aantal malen per seconde. door onze hersentraagheid zien we beelden (vergelijk film).
Stel voor dat de lichtgevoelige cel ergens op het scherm staat ter plekke van een geprint blokje. Als de electronenbundel het beeld begint te schrijven (dit kunnen we dus testen) is de tevoren geinitialiseerde VIA-counter begonnen met aftellen.
Als ter plekke van de lichtcel een blokje wordt geschreven krijgt de cel licht binnen (ook te testen) en lezen we de counter uit. Deze waarde gaat naar de variabele L. Na wat rekenwerk hebben we de positie.
De 1 MHz van de Atom blijkt te langzaam om ook horizontaal de plaats te bepalen.



```
  1 G.10
  2 ***************************************
  3 *** UITLEESROUTINE LICHTPEN         ***
  4 *** RESULTAAT IN BASIC-VARIABELE L  ***
  5 *** DOOR JOOST DE WIJS              ***
  6 *** RAPPORTSTRAAT 12-A              ***
  7 *** 5504 BP  VELDHOVEN              ***
  8 ***************************************
  9
 10 DIMLL4;F.I=0TO4;LL(I)=-1;N.
 20 F.I=1TO2;P=#2800;P.$21;[
 30 :LL0 BIT #B800;BPL LL0              \LICHT ?
 40 :LL1 BIT #B002;BPL LL1               \GEEN FLYBACK ?
 50 :LL2 BIT #B002;BMI LL2              \FLYBACK ?
 60 LDA@0;STA#B808;STA#B809;STA#B802 \COUNTERS MAX.;VIA INPUT
 70 :LL3 BIT #B002;BPL LL3              \GEEN FLYBACK (LINKSBOVEN)?
 80 :LL4 BIT #B800;BPL LL4              \LICHT ?
 90 LDA #B808:STA #032D                 \COUNTERS IN L
100 LDA #B809;STA #034B                 \
110 RTS                                 \TERUG NAAR BASIC
120 ];P.$6;N.
130 L=0
135
140 REM *** 192 VERTICALE LIJNEN ***
150 P.$12;?225=0
160 F.I=0TO512 STEP32;I?#8000=127;N.
170 LINK#2800
180 P.$30'''''$9$9$9$9$9$9$9,(61050-L)/63
190 GOTO 170
200
240 REM *** 16 KEUZE'S IN TEKSTMODE ***
250 REM *** BV IN MENU              ***
260 P.$12;?225=0;P.$9$9"=EINDE TEST"
270 F.I=0TO512 STEP32;I?#8000=76
280 IFI<310;I?#8001=((I/32)+48);N.
290 I?#8001=((I/32)-9);N.
300 @=0
310 LINK#2800
320 L=(60750-L)/725
330 IF L<0 OR L>15;G.310
340 IF L=0:END
350 P.$30'''''$9$9$9$9$9$9$9,&L,$7
360 GOTO 310
```

Inleiding

Binnen de Acorn club zijn diverse typen printers in gebruik. Al deze printers hebben d.m.v. stuurcodes omschakelbare mogelijkheden zoals: Condensed en Enlarged lettertypes, Tab-setting, Bit image e.d.
Voor dezelfde functie kunnen de stuurcodes per printer verschillen. Dit heeft tot gevolg dat een programma of een wordpack file gemaakt voor de ene printer niet goed werkt op andere printers. We moeten de stuurcodes bijstellen, en om dit te kunnen doen moeten we weten wat bij de diverse printers de overeenkomende stuurcodes zijn. Dit overzicht kan u hierbij van dienst zijn. Het geeft per printer aan voor welke functie men welke bijbehorende stuurcode moet gebruiken.

De Printers

De in dit overzicht betrokken printers zijn:

ITT	3351
EPSON	MX80
	RX80
	MX100
STAR	DP510
	DP515
	Gemini 10 X
MICROLINE	80
NEC	PC8023
OLIVETTI	Inktjet PR2300

De Codes

Alle stuurcodes hebben gemeen dat ze met een ASCII control code beginnen.

b.v. SI (#0F).

Veel codes bestaan uit een ESC (#1B) gevolgd door een letter

b.v. ESC A

Dit wordt naar de printer gestuurd als de getallen #1B en #41 (decimaal 27 en 65). In de wordpack zou dat er als volgt uit kunnen zien: .o27.o65
Er zijn ook codes waarbij u een of meer getallen moet invoeren. B.v. een variabele line-feed. Dit zou kunnen zijn:

ESC A n

waarbij n een getal is dat voor n/72 inch line feed zorgt. Als ik 15/72 inch line feed zou willen hebben zou dit dus in de textverwerker er als volgt uit kunnen zien:

.o27.o65.o15

Met getallen als n dient u er om te denken dat de getallen 2, 3 en het getal in #FE (normaal dec. 10) niet naar de printer verstuurd worden.

In dit overzicht zal ik het getal n niet in het overzicht opnemen. Dus als bij uw printer een line feed van n/72 inch als: ESC A n moet worden ingevoerd, staat in de tabel alleen ESC A. Ditzelfde geldt voor de zgn "afsluiters" b.v. bij tab-setting om aan te geven dat er niet meer tabs komen.
Voorbeeld: een tab op de 10e, 20e en 30e positie zou als volgt kunnen worden ingevoerd:

.o27.o72.o10.o20.o30.o0
ESC D '10 '20 '30 Afsluiter

Toch wordt in de tabel alleen ESC D opgegeven. Dit is gedaan om de tabellen niet te onoverzichtelijk te maken.

Ook nog een opmerking m.b.t. de zgn "graphics modes". De getallen achter de stuurcodes hebben een dermate andere betekenis dat alleen het verwissellen van de code niet werkt. We kennen de bit-image of dot-image mode, waarbij men direct de printpennen aanstuurt. Elke pen heeft een bepaalde waarde (1 2 4 8 e.d.), en door het versturen van het juiste getal kan men bereiken dat b.v. de 1e, 2e en 5e pen een afdruk maken. Een andere methode is de zgn semi-graphics mode. Dit zijn in de printer "ingebakken" graphics symbolen die vaak met een stuurkode boven de #7F te bereiken zijn. Per printer is dit alles dus anders. Enige voorbeelden:

Epson,
ITT3351 en : Bit image ; ESC K(L) l h
Star waarbij: l = aantal bit image char. op regel % 256
h = aantal bit image char. op regel / 256

Microline 80 : Block graphics boven de # 7F

NEC PC8023 : Bit image ; ESC S l h
voor l en h zie EPSON MX80
bovendien ook nog blockgraphics

Olivetti : ESC G b h d s Z
waarbij : b = beginpositie vanaf zijkant
h = breedte
d = diepte
s = spatie tussen 2 lijnen

De in het overzicht gebruikte codes hebben de volgende waarden:

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NULL	30	1E	RS	60	3C	<
1	01	SOH	31	1F	US	61	3D	=
2	02	STX	32	20		62	3E	>
3	03	ETX	33	21	!	63	3F	?
4	04	EOT	34	22	"	64	40	@
5	05	ENQ	35	23	#	65	41	A
6	06	ACK	36	24	$	66	42	B
7	07	BEL	37	25	%	67	43	C
8	08	BS	38	26	&	68	44	D
9	09	HT	39	27	'	69	45	E
10	0A	LF	40	28	(	70	46	F
11	0B	VT	41	29	)	71	47	G
12	0C	FF	42	2A	*	72	48	H
13	0D	CR	43	2B	+	73	49	I
14	0E	SO	44	2C	,	74	4A	J
15	0F	SI	45	2D	-	75	4B	K
16	10	DLE	46	2E	.	76	4C	L
17	11	DC1	47	2F	/	77	4D	M
18	12	DC2	48	30	0	78	4E	N
19	13	DC3	49	31	1	79	4F	O
20	14	DC4	50	32	2	80	50	P
21	15	NAK	51	33	3	81	51	Q
22	16	SYN	52	34	4	82	52	R
23	17	ETB	53	35	5	83	53	S
24	18	CAN	54	36	6	84	54	T
25	19	EM	55	37	7	85	55	U
26	1A	SUB	56	38	8	86	56	V
27	1B	ESC	57	39	9	87	57	W
28	1C	FS	58	3A	:	88	58	X
29	1D	GS	59	3B	;	89	59	Y
						90	5A	Z

Print mode

	ITT 3351 Epson: MX80 MX100	Epson: RX 80	Star: DP510 DP515	Star: Gemini-10 X	Nec: PC8023	Microline 80	Olivetti PR2300-inktjet
Condensed On Condensed Off	SI DC2	SI DC2	SI DC2	SI DC2	ESC Q	GS	
Double Highth							ESC '
Double Strike (Enhanced) Off	ESC G ESC H	ESC G ESC H	ESC G ESC H	ESC G ESC H	ESC ! ESC "		
Elite On Elite Off	ESC M DC2	ESC M ESC P		ESC B STX			
Emphasised On Emphasised Off	ESC E ESC F	ESC E ESC F	ESC E ESC F	ESC E ESC F			
Enlarged On Enlarged Off		ESC W SOH ESC W NUL		ESC W SOH ESC W NUL		US	ESC 3
Enl. line On Enl. line Off	SO SI	SO SI	SO SI	SO SI	SO SI		
Greek characters					ESC &		
Italics On Italics Off		ESC 4 ESC 5	ESC 4 ESC 5	ESC 4 ESC 5			
Normal				DC4	ESC N	RS	ESC <
Proportional					ESC P		
Subscript On Subscript Off	ESC S NUL ESC T 1)	ESC S NUL ESC T	ESC S NUL ESC T NUL	ESC S NUL ESC T NUL			
Superscript Superscript Off	ESC S SOH ESC T 1)	ESC S SOH ESC T	ESC S SOH ESC T SOH	ESC S SOH ESC T SOH			
Underlined: Normal On Normal Off Double Spickled	ESC - 1)	ESC - SOH ESC - NUL	ESC -	ESC - SOH ESC - NUL	ESC X ESC Y		ESX # 0 ESC # 1 ESC # 2

Line spacing

	ITT 3351 Epson: MX80 MX100	Epson: RX 80	Star: DP510 DP515	Star: Gemini-10 X	Nec: PC8023	Microline 80	Olivetti PR2300-inktjet
Line-feed 1/3 inch					ESC \ LF		
1/6 inch	ESC 2	ESC 2	ESC 2	ESC 2	ESC A	ESC 6	ESC E
1/8 inch	ESC 0	ESC 0	ESC 0	ESC 0	ESC B	ESC B	ESC F
7/72 "		ESC 1	ESC 1	ESC 1			
n/72 "	ESC A	ESC A	ESC A	ESC A			
n/144 "			ESC 3	ESC 3	ESC T		
n/216 "		ESC 3					
1 cm					ESC \ DC2		
1 time LF n lines				ESC a	US		
1 time LF n/72 inch	ESC J		ESC J				
n/144 inch				ESC J			
n/216 inch					ESC J		
Forward LF direct.					ESC f		
Reversed LF dir.					ESC r		

Format Control

	ITT 3351 Epson: MX80 MX100	Epson: RX 80	Star: DP510 DP515	Star: Gemini-10 X	Nec: PC8023	Microline 80	Olivetti PR2300-inktjet
Set HTab	ESC D	ESC e NUL	ESC D	ESC D	ESC <		ESC P
Reset single HTab					ESC >		
Clear all Htabs					ESC 2		
1 time Htab n kol				ESC b			
Tab n char.		ESC f NUL					
Set Vtab	ESC B	ESC e SOH	ESC P	ESC P	GS		
Tab n lines		ESC f SOH					
Set Left Marigin		ESC l	ESC M	ESC M	ESC L		
Set Right Marigin		ESC Q	ESC Q	ESC Q			
Form length:							
n lines	ESC C	ESC C	ESC C	ESC C			
n inch	ESC C NUL	ESC C NUL	ESC C NUL	ESC C NUL			
Paper width:							
64 coloums						ESC B	
80 coloums			ESC B SOH			ESC A	
96 coloums			ESC B STX				
132 coloums			ESC B ETX				
n coloums	ESC Q						

Bit Image and semi-graphics

	ITT 3351 Epson: MX80 MX100	Epson: RX 80	Star: DP510 DP515	Star: Gemini-10 X	Nec: PC8023	Microline 80	Olivetti PR2300-inktjet
Single density	ESC K	ESC K		ESC K	ESC S		ESC G
Dual density	ESC L	ESC L		ESC L			
Quadruple density		ESC Z		ESC z			
Fast dual density		ESC Y		ESC y			
CRT Bit image		ESC * EOT					
CRT II Bit image		ESC * ACK					
Inverse Graphics							ESC -
2 x zoom graphics							ESC /
Repeat dot string					ESC V		
Semigraphics		ESC m			ESC #		

Miscellaneous

	ITT 3351 Epson: MX80 MX100	Epson: RX 80	Star: DP510 DP515	Star: Gemini-10 X	Nec: PC8023	Microline 80	Olivetti PR2300-inktjet
Bidirectional uni-directional	ESC U NUL USC U SOH 1)	ESC U NUL ESC U SOH	ESC U NUL ESC U SOH	ESC U NUL ESC U SOH	ESC (ESC)		
Unidirectional line		ESC <					
Skip over perf. cancel skip-over	ESC N ESC O	ESC N ESC O	ESC N ESC O	ESC N ESC O			
Logical seeking mode Incremental mode					ESC] ESC [		
Select paper end Cancel p.e. det.	ESC 9 ESC8	ESC 9 ESC 8		ESC 9 ESC 8			
Select printer Deselect printer Reset printer	DC1 DC3 ESC @ 1)	DC1 DC3 ESC @		DC1 DC3 ESC @	DC1 DC3 ESC \ DC4		ESC 0
Repeat char. n times					ESC R		
Half speed Full speed		ESC s SOH ESC s NUL					
Select bell code Cancel bell code				ESC $ NUL ESC $ SOH			

Opmerkingen: 1) Alleen Epson

INTIKKEN EN RUNNEN MAAR

TEKENEN

```
 10 PROGRAM TEKENEN
 20
 30 ?#23=0;?#24=#30
 40 DIM A1000,B1000,S3
 50
 60 PROC STAP
 70  PLOT 14,I,J;PLOT 14,X,Y
 80  I=X;J=Y
 90 PEND
100
110 PROC SCHUIF
120  P=X;Q=Y;I=X;J=Y
130 PEND
140
150 PROC TEKEN
160  IFE)998 OR (X=P AND Y=Q);G.a
170  MOVE P,Q;PLOT 6,X,Y
180  INKEY T
190  MOVE P,Q;PLOT 6,X,Y
200  IF T=CH"C" G.a
210  MOVE P,Q;DRAW X,Y
220  A?E=P;B?E=Q
230  A?(E+1)=X;B?(E+1)=Y
240  E=E+2;B?E=#FF;SCHUIF
250aPEND
260
270 PROC VERWIJDER,N,I,J,X,Y
280  N=1;DO
290   I=A?N;J=B?N
300   X=A?(N+1);Y=B?(N+1)
310   DO KNIPPER(I,J,X,Y)
320   KEY T;UNTIL T)0
330   IF T=CH"C" WEG(N);N=N-2
340  N=N+2;UNTIL N=E OR T=CH"."
350 PEND
360
370 PROC KNIPPER(I,J,X,Y),T
380  MOVE I,J;PLOT 6,X,Y
390  FOR T=1 TO 6;WAIT;N.
400  MOVE I,J;PLOT 6,X,Y
410  FOR T=1 TO 6;WAIT;N.
420 PEND
430
440 PROC WEG(N)
450  ICOPY(A+N+2),(A+E),(A+N)
460  ICOPY(B+N+2),(B+E),(B+N)
470  MOVE I,J;PLOT 6,X,Y
480  E=E-2;N=N-2
490 PEND
500
510 PROC UITTEKENEN
520  DO P=A?E;Q=B?E
530   X=A?(E+1);Y=B?(E+1)
540   MOVE P,Q;DRAW X,Y
550   E=E+2
560  UNTIL B?E=#FF
570 PEND
580
590 P.$12"TEKENPUNT VERPLAATSEN"'
600 P."MET A,S,W EN Z"''
610 P."Move (ONTHOUDT POSITIE)"''
620 P."Draw (CORRECTIE MET C)"''
630 P."VERWIJDEREN: V EN C"''
640 IN."GEGEVENS GELADEN"$S
650 E=1;CLEAR4
660 IF $S="J" UITTEKENEN
670 X=0;Y=0;I=0;J=0;P=0;Q=0
680
690 DO KEY T;?S=T;S?1=#D
700 CASE $S OF
710("A") X=X-1;STAP
720("S") X=X+1;STAP
730("W") Y=Y+1;STAP
740("Z") Y=Y-1;STAP
750("M") SCHUIF
760("D") TEKEN
770("V") VERWIJDER
780 CEND
790 UNTIL 0
```

VOOR DE GAGSROM

```
  5REM GAGSBOX NODIG
 10BASE#2D
 20DEF SPINAA,0000001111000000
 30DEF:       0000011111100000
 40DEF:       0000111001110000
 50DEF:       1000111001110001
 60DEF:       1000100000010001
 70DEF:       0100111001110010
 80DEF:       0010011111100100
 90DEF:       0001000110001000
100DEF SPINAB,1100111111110011
110DEF:       0011111111111100
120DEF:       0000011111100000
130DEF:       1111101111011111
140DEF:       0000011111100000
150DEF:       0011101111011100
160DEF:       1100000110000011
170DEF:       0000001001000000
180
190CLEAR4;NOSNOW
200FORN=1 TO 360 S.30
210%X=COSRADN;%Y=SINRADN
220MOVE%(11*%X+128),%(11*%Y+97)
230DRAW%(150*%X+128),%(150*%Y+97)
240N.
250
260FORN=11 TO 90 S.15
270CIRCLE1,128,97,N
280N.
290
300SET:SPINAA,77,192
310SET:SPINAB,77,184
320ASSIGN SPINAA,1
330ASSIGN SPINAB,2
340H=1;V=-2
350
360FORS=1 TO 43
370SHOVE1,H,V;SHOVE2,H,V
380FORU=1 TO S*3;N.
390N.
400END;        P.E.
```

SPACE

```
 10 PROGRAM SPACE
 20 REM R. VAN DRUNEN
 30 P.$12'
 40 P."** DISK MAP **"'"INSERT DISK & PRESS RETURN"'
 50 LINK #FFE3
 60 *DIR
 70@=0;P.$12
 80M=#FFFF;U=#9C
 90L=#2000;H=#2100
100N=H?5
110P.'"ENTRIES :"N/8"          "
120S=(H?6&15)*256+H?7
130P."SECTORS :"S
140P.'"  no  name     ssec sec   length "'
150E=N;Z=1;Y=0;G=0
160F=2
170d
180 B=H!E&M;D=H!(E+2)&M;W=H!(E+4)&M+(H?(E+6)/16)*M
190T=(H?(E+6)&15)*256+H?(E+7)
200R=W/256;IFW&255()0;R=R+1
210IFF)=T G.240
220IFF(T;@=5;P."  -   blank   "F T-F';A=F;C=T-1;Y=Y+1
230 G=G+(T-F);F=C+1;G.240
240IFE=0;G.r
250@=3;P.Z"   ";@=5
```

```
260FOR O=0TO6
270P.$(O?(L+E));N.
280P. T,R;@=8;P.&W'
290 A=T ;C=T+R-1
300IF R=0;G.320
310F=C+1
320 E=E-8;Z=Z+1
330 IF(Z+Y)%11=0;LINK #FFE3;!#DE=#8080
340 IFE)=0;G.d
350rDO P." "
360 UNTIL !#DE=#81E0;!#DE=!#DE-#20
370 @=0
380 P."BLANK SECT:"G';P."FREE SPACE :"G/4" "G%4"/4 K"
390LINK #FFE3;P.$12;G.110
```

BREUKEN

```
 10 REM BREUKEN
 20 REM TINY VERSCHUREN
 30 @=0;P.$12"    werken met breuken"''
 40 P."VOORBEELD : 1 3/4 + 2/7"'
 50 P."INVOEREN ALS VOLGT :"''
 60 P."BREUK 1"'
 70 P."GEHEEL GETAL    1"'
 80 P."DELER           3"'
 90 P."NOEMER          4"''
100 P."BREUK 2"'
110 P."GEHEEL GETAL    0"'
120 P."DELER           2"'
130 P."NOEMER          7"''
140 P."DRUK EEN TOETS ";LI.#FFE3
150aP.$12"voer de breuken in"'''
160 P."BREUK 1"'"-------"'
170 IN."GEHEEL GETAL    "A
180 IN."DELER           "B
190 IN."NOEMER          "C
200 P.'"BREUK 2"'"-------"'
210 IN."GEHEEL GETAL    "D
220 IN."DELER           "E
230 IN."NOEMER          "F
240 L=SGNA;O=SGND;A=ABSA;D=ABSD
250 IFL=0;L=1
260 IFO=0;O=1
270 X=C;Y=F;P=X*Y
280bZ=X-Y*(X/Y);X=Y;Y=Z
290 IF Z G.b
300 Y=ABS(P/X)
310 P.$12"KLEINSTE GEMENE VEELVOUD = "Y''
320 P."BREUK 1 :"'
330 G=(C*A+B)*(Y/C)*L
340 P.A*L" "B"/"C" = "(C*A+B)*L"/"C" = "G"/"Y'
350 P."BREUK 2 :"'
360 H=(F*D+E)*(Y/F)*O
370 P.D*O" "E"/"F" = "(F*D+E)*O"/"F" = "H"/"Y''
```

```
380 P."1 = OPTELLEN"'
390 P."2 = AFTREKKEN"'
400 P."3 = VERMENIGVULDIGEN"'
410cP."4 = DELEN         "
420 IN."KEUZE "K
430 IF K<1 OR K>4 P.$11;G.c
440 IF K=1 X=G+H;G.d
450 IF K=2 X=G-H;G.d
460 IF K=3 X=G*H;Y=Y*Y;G.d
470 IF K=4 X=G*Y;Y=H*Y
480dP.'"TOTAAL :"'X"/"Y
490 L=SGNX;X=ABSX;M=X;N=Y
500eZ=X-Y*(X/Y);X=Y;Y=Z
510 IF Z G.e
520 I=M/X;J=N/X
530 IF X>1 P." = "I*L"/"J
540 IF I<J G.g
550 S=I/J;T=I-S*J
560 P." = "S*L" ";IF T=0 G.g
570 P.T"/"J
580gP.''
590hIN."WILT U NOG MEER <J/N> "$V
600 IF $V="J" G.a
610 IF $V="N" G.f
620 P.$11;G.h
630fP.$12''''''"  SUCCES VERDER, EN TOT ZIENS."';E.
```

KALENDER

```
 10 REM ***********************
 20 REM *                     *
 30 REM *** K A L E N D E R ***
 40 REM * NAAR EEN PROGRAMMA  *
 50 REM * VAN JOS NEERPELT IN *
 60 REM * 'HOBBIT,DEC.1982'   *
 70 REM * BEWERKT VOOR ACORN  *
 80 REM * DOOR G.DE GRAAF     *
 90 REM * CEINTUURBAAN 70     *
100 REM * 1403AD BUSSUM       *
110 REM *                     *
120 REM ***********************
130 REM SUPER-BASIC IN VOETJE 2
140?#BFFF=2;LINK#AF04
150DIMG(21),N(124),O(47),U(13),I(20),@(1)
160GOS.720
170$N=[$N+$O+$U]
180P.'''"WENST U DE KALENDER VAN EEN BEPAALD"
190P.'"JAAR NA 1582, DRUK DAN EEN TOETS IN."
200P.'''"VAN WELK JAAR WENST U EEN KALENDER";IN.J
220IF J<1582G.760
230P.$12;D=1;M=1;S=112
240P.$12"          *** KALENDER ***"'" "'
250GOS.810 ;P."  JANUARI "J" "';GOS.570 ;GOS.580 ;P." "'
260J=J+1
270GOS.810 ;P." FEBRUARI "J" "';M=2
280T=J%4
290IF T=0S=106;G.310
300S=103
```

```
310GOS.570 ;GOS.580 ;P." "'
320J=J+1
330GOS.810 ;P."    MAART "J" "';M=3;S=112;GOS.570 ;GOS.580
340P." "'
350GOS.770
360GOS.810 ;P."    APRIL "J" "';M=4;S=109;GOS.570 ;GOS.580
370P." "'
380GOS.810;P."     MEI "J" "';M=5;S=112;GOS.570 ;GOS.580
390P." "'
400GOS.810;P."    JUNI "J" "';M=6;S=109;GOS.570 ;GOS.580
410P." "'
420GOS.770
430GOS.810 ;P."     JULI"J" "';M=7;S=112;GOS.570 ;GOS.580
440P." "'
450GOS.810 ;P." AUGUSTUS "J" "';M=8;S=112;GOS.570
460GOS.580 ;P." "'
470GOS.810 ;P."SEPTEMBER "J" "';M=9;S=109;GOS.570
480GOS.580 ;P." "'
490GOS.770
500GOS.810 ;P."  OKTOBER "J" "';M=10;S=112;GOS.570
510GOS.580 ;P." "'
520GOS.810;P." NOVEMBER "J""';M=11;S=109;GOS.570
530GOS.580 ;P." "'
540GOS.810 ;P." DECEMBER "J""';M=12;S=112;GOS.570
550GOS.580 ;P." "'
560END
570P."         .MA.DI.WO.DO.VR.ZA.ZO "';R.
580A=(J*365)+D+31*(M-1)
590IF M<3G.610
600A=A-%(.4*M+2.3);G.620
610J=J-1
620F=A+%(J/4)-%(.75*(1+%(J/100)))
630E=F-(7*%(F/7))
640IF E<2E=E+5;G.660
650E=E-2
660L=19-3*E
670F.X=L TO S S.21
680Y=21
690IFX+21>S Y=S-X
700P."          "MID$($N,X,Y)," "'
710N.;R.
720$N="...................1..2..3..4..5..6..7..8..9.10.11.12"
730$O=".13.14.15.16.17.18.19.20.21.22.23.24.25.26.27.28."
740$U=".29.30.31......"
750R.
760P.$3"VERKEERD JAAR!!!"'
770P.$3"WENST U HET VOLGENDE TRIMESTER J/N ?";IN.$0
780IF$0="J"G.800
790E.
800R.
810P.TAB(10);R.
830 REM OM DE KALENDER OP PAPIER
840 REM TE PRINTEN MOETEN DE VOLGENDE
850 REM REGELS TOEGEVOEGD WORDEN,
860 REM ALTHANS VOOR EEN MICROLINE 80.
870 REM     210 P.$2 OF $2$29 OF $2$31
880 REM     765 P.$2
890 REM     775 P.$2
```

BRAILLE

```
 10?#BFFF=2;LINK#AF04;REM AAN-ROEP SUPER-BASIC
 12REM ALS JE DEZE NIET BEZIT
 13REM DAN REGEL 10 WEGLATEN
 14REM EN DE REGELS 700 T"M 750
 15REM VERANDEREN IN BIJV.P.$7
 20 P.$12;?#E1=0
 30 P."BRAILLE-TRAINER"'
 40 P."''''''''''''''''"'
 50 P.'"EERST EEN KLEINE TOELICHTING OP"'
 60 P."HET BRAILLE-PROGRAMMA."''
 70 P."BRAILLE IS HET SCHRIFT DAT DOOR"'
 80 P."BLINDEN WORDT GEBRUIKT."'
 90P. '"ELKE LETTER BESTAAT UIT EEN COM-"'
100 P."BINATIE VAN TEN HOOGSTE 6 PUNTEN."''  "druk toets"
110 LI.#FFE3
120P.'''"ALS VOLGT GRANGSCHIKT:"''
130P."          . ."'"         . ."'"        . ."''
140 P."' '"'"' '"'"' '"'"      "$11$11" =A"'''
150 P."' '"'"' '"'"' '"'"      "$11$11" =B"'''
160P. "druk toets"';LI.#FFE3
170P.''
180 P."  '"'"  '"'"' '''"DIT IS HET CIJFERTEKEN."'
190 P."HET GETAL 1 IS HET CIJFERTEKEN + A."'
200 P."HET GETAL 2 IS HET CIJFERTEKEN + B."''
210 P."HET GETAL 83 BIJV. WORDT:"'
220 P."  '  '   '"'"  '  '"'"' '"''
225P."                "$11$11$11'"          CIJFERTEKEN +H +C"'
230P."druk toets"';LI.#FFE3
240P.''''''
250P."LEESTEKENS:"''
260P."''  ''  ''  ''  ''  ''  ''  ''"'
270P."''  ''  ''  ''  ''  ''  ''  ''"'
280P."''  ''  ''  ''  ''  ''  ''  ''"'
290P.''"  ,   ;   :   .   ?   !   (   )  "''''
292P."JE KUNT NU HET ALFABET OEFENEN."'
293P."DE GELUIDJES HEBBEN SLECHTS "'
294P."LUDIEKE (SPEELSE) WAARDE."''
300 F.I=1TO 60;WAIT;N.
310DIMA1,B1,C1,D1,E1,F1,G1,H1,I1
320DIMJ1,K1,L1,M1,N1,O1,P1,Q1,R1,S1
330DIMT1,U1,V1,W1,X1,Y1,Z1
340 P.$21
350 P=#80;[;LDA@B2;STA#208;LDA@254;STA#209;RTS;]
360 Z=#FE940080;!#208=Z
370F.I=0 T.7;P.$32;N.;P.$9
380P.$6
390 IN.$A
400P.$32$8$11$32$A'
410IF$A="A"     P."'  "'';GOS.a
420 IF $A="B"P."   "'"'   "'';GOS.a;GOS.b
430 IF $A="C"P."   "'';GOS.a;GOS.d
440 IF $A="D"P."   '"'"   '"'';GOS.a;GOS.d;GOS.e
450 IF $A="E"P."'   "'"   '"'';GOS.a;GOS.e
460 IF $A="F"P."'  '"'"'   "'';GOS.a;GOS.b;GOS.d
470 IF $A="G"P."'  '"'"'   "'';GOS.a;GOS.b;GOS.d;GOS.e
480 IF $A="H"P."   "'"'   "'';GOS.a;GOS.b;GOS.e
```

```
490 IF $A="I"P."  `"'"`  "'';GOS.d;GOS.b
500 IF $A="J"P."  `"'"` `"'';GOS.b;GOS.d;GOS.e
510 IF $A="K"P."`  "'"   "'"`  "'';GOS.a;GOS.c
520 IF $A="L"P."`  "'"`  "'"`  "'';GOS.a;GOS.b;GOS.c
530 IF $A="M"P."` `"'"   "'"`  "'';GOS.a;GOS.c;GOS.d
540 IF $A="N"P."` `"'"  `"'"`  "'';GOS.a;GOS.c;GOS.d;GOS.e
550 IF $A="O"P."`  "'"  `"'"`  "'';GOS.a;GOS.c;GOS.e
560 IF $A="P"P."` `"'"`  "'"`  "'';GOS.a;GOS.b;GOS.c;GOS.d
570IF$A="Q"P."` `"'"` `"'"` "'';GOS.a;GOS.b;GOS.c;GOS.d;GOS.e
580 IF $A="R"P."`  "'"` `"'"`  "'';GOS.a;GOS.b;GOS.c;GOS.e
590 IF $A="S"P."  `"'"`  "'"`  "'';GOS.b;GOS.c;GOS.d
600 IF $A="T"P."  `"'"` `"'"`  "'';GOS.b;GOS.c;GOS.d;GOS.e
610 IF $A="U"P."`  "'"   "'"` `"'';GOS.a;GOS.c;GOS.f
620 IF $A="V"P."`  "'"`  "'"` `"'';GOS.a;GOS.b;GOS.c;GOS.f
630 IF $A="W"P."  `"'"` `"'"  `"'';GOS.b;GOS.d;GOS.e;GOS.f
640 IF $A="X"P."` `"'"   "'"` `"'';GOS.a;GOS.c;GOS.d;GOS.f
650IF$A="Y"P."` `"'"  `"'"` `"'' ;GOS.a;GOS.c;GOS.d;GOS.e;GOS.f
660 IF $A="Z"P."`  "'"  `"'"` `"'';GOS.a;GOS.c;GOS.e;GOS.f
670IF$A="1"P."  `  `  "'"  `   "'"`  `     "'';GOS.g;GOS.a
671IF$A="2"P."  `  `  "'"  `  `"'"` `"';GOS.g;GOS.a;GOS.b
672IF$A="3"P."  `  ``"'"  `   "'"` `"';GOS.g;GOS.a;GOS.d
673IF$A="4"P."  `  ``"'"  `    `"'"` `"';GOS.g;GOS.a;GOS.d
674 IF$A="4"GOS.e
680G.340
685 N.
690END
700aBEEP205,13;R.
710bBEEP183,13;R.
720cBEEP137,13;R.
730dBEEP102,13;R.
740eBEEP81 ,13;R.
750fBEEP68 ,13;R.
751gBEEP205,3;BEEP183,3;BEEP163,3;BEEP154,3
752F.I=1TO30;WAIT;N.;R.
760 REM :                   :
761 REM :   G.de GRAAF      :
762 REM :   CEINTUURBAAN 70 :
763 REM :   1403AD BUSSUM   :
764 REM -------------------
```

TEMPERATUUR

```
 10 PROGRAM TEMPERATUUR
 20
 30 REM TINY VERSCHUREN
 40
 50 DIMS9
 60 $S="DANK U"
 70
 80 PROC MENU(:A)
 90  P.$12''
100  HTAB5;P."WELKE WAARDE IS BEKEND"'''
110  HTAB6;P."CELCIUS      =      1"''
120  HTAB6;P."REAUMUR      =      2"''
130  HTAB6;P."FARHENHEIT   =      3"''
140  HTAB6;P."KELVIN       =      4"'''
150  HTAB5;IN."KEUZE "A
160  IF A<1 OR A>4;P.$11;G.150
170 PEND
```

```
180
190 PROC CELSIUS
200  FIN."CELCIUS "%C
210  %R=4/5*%C;%F=9/5*%C+32;%K=%C+273
220 PEND
230
240 PROC REAUMUR
250  FIN."REAUMUR "%R
260  %C=5/4*%R;%F=9/4*%R+32;%K=5/4*%R+273
270 PEND
280
290 PROC FAHRENHEIT
300  FIN."FARHENHEIT "%F
310  %C=5/9*(%F-32);%R=4/9*(%F-32);%K=5/9*%F+255.2
320 PEND
330
340 PROC KELVIN
350  FIN."KELVIN "%K
360  %C=%K-273;%R=4/5*(%K-273);%F=9/5*(%K-255.2)
370 PEND
380
390 PROC FIXPR(%V),@,V
400  V=%(%V*100+SGN%V*0.5)
410  XIF V<0 AND V>-100;P."      -0"
420  ELSE P.V/100
430  @=0;V=ABSV%100;P."."V/10,V%10
440 PEND
450
460 REM MAIN PROGRAM
470
480 DO
490 MENU(K)
500 P.$12''"     VUL DE WAARDE IN GRADEN IN"'';HTAB10
510 CASE K OF
520 (1) CELSIUS
530 (2) REAUMUR
540 (3) FAHRENHEIT
550 (4) KELVIN
560 CEND
570  P.$12''"ONDERSTAANDE WAARDEN ZIJN GELIJK"'''
580 FIXPR(%C);P." GRADEN CELSIUS"''
590 FIXPR(%R);P." GRADEN REAUMUR"''
600 FIXPR(%F);P." GRADEN FAHRENHEIT"''
610 FIXPR(%K);P." GRADEN KELVIN"''
620 INKEY A
630 UNTIL A=CH"E";P.$12
640 FOR K=0 TO 6;L=2*K+6
650  VTAB(L);HTAB(L);P.$S?K
660 N.
670 END
680
690 P.S. ALS MEN EEN VEELZIJDIGER
700 FIXPRINT WIL MET WILLEKEURIG
710 AANTAL POSITIES ACHTER DE KOMMA,
720 ONDERDRUKKING VAN ONINTERESSANTE
730 NULLEN EN NORMAAL GEBRUIK VAN @,
740 DAN GEBRUIKE MEN ONDERSTAANDE
750 PROCEDURE.
```

```
760
770 PROC FIXPR(%X,F),@,Z
780  @=@-F-1;F=%(10↑F+0.5);%X=%X+0.5/F*SGN%X
790  FIF%X(0FIF%X)-1;FORZ=1TO @-2;P." ";N.;P."-0.";G.a
800  P.%X"."
810a Z=%(ABS(%X-%%X)*F)
820  @=1;DOF=F/10;P.Z/F;Z=Z%F;U.Z=0
830  IFF)1;DOP." ";F=F/10;U.F=1
840 PEND
```

ANAGRAM

```
100 PROGRAM ANAGRAM
110
120 PROC EXCHANGE(X,T),B
130  B=T?X
140  T?X=?T
150  ?T=B
160 PEND
170
180 PROC COMBINATIONS(S),I,M
190  FOR I=1 TO LEN(S)
200   M=1
210   DO
220    EXCHANGE (M,S)
230    M=M+1
240    PRINT$S'
250   UNTIL M=LEN(S)
260  NEXTI
270 PEND
280
290 PROC ENTRY,H
300  PRINT$12"          ANAGRAMS"
310  PRINT  "           --------"''
320  H=#4000
330  INPUT"ENTER STRING :"$H
340  PRINT''"POSSIBLE COMBINATIONS ARE :"''
350  COMBINATIONS(H)
360 PEND
370
380 REM MAIN PROGRAM
390 ENTRY
400 END
410
420 Programma in P-charme om alle lettercombinaties uit
430 een ingevoerde string samen te stellen
440 E. Snelders
450 Welhaak 63
460 9932 BK Delfzijl
```

```
>ANAGRAM
          ANAGRAMS
          --------

ENTER STRING :?ATOM


POSSIBLE COMBINATIONS ARE :

TAOM
OATM
MATO
AMTO
TMAO
OMAT
MOAT
AOMT
TOMA
OTMA
MTOA
ATOM
```

In Acorn Nieuws 2 van 1984 plaatsten wij een overzicht van verschenen literatuur over de Atom. Onlangs is er bij uitgeverseij Elektuur B.V. een boekje verschenen dat ook in genoemd overzicht thuis hoort. Omdat het werkje mijzelf erg aansprak wil ik hier graag een overzicht geven van de inhoud.

PROGRAMMA'S VOOR DE ACORN ATOM
(Erik Spin & Klaus Siol)

Het boek is verdeeld in vier hoofdstukken en heeft drie appendices. Ik zal elk kort beschrijven.

* Programma's

Dit beslaat het grootste deel van het boek. Er worden hierin achttien listings van voornamelijk spelletjes gegeven. Elk is voorzien van een uitgebreide uitleg van de werking. (Als alle copy voor Acorn Nieuws zo goed gedocumenteerd was ..) De programma's zijn gerangschikt naar oplopende moeilijkheidsgraad. Vaak worden ook de in vorige programma's besproken routines weer benut. Het voordeel is duidelijk, er kan op die mannier een tamelijk complex programma worden gebouwd zonder dat de iets minder ver gevorderden onder ons behoeven af te haken. Een voorbeeld van de werkwijze is het volgende:
Er wordt begonnen met uitleg te geven over de werking van de standaard plotroutines. Vervolgens wordt er met behulp daarvan een soort spritehandler gemaakt. En tenslotte maken het grootste deel van de verdere programma's gebruik van deze software. Zo leer je ongemerkt heel wat over de grafische mogelijkheden van de Atom.

* Grapjes met graphics

Hierin vinden we voornamelijk korte listings. Voorbeelden zijn: een stippellijn routine, een fraaie explosie en een logarithmische schaal.

* Tips

Dit is ook een zeer interressant hoofdstukje. Veel van de programma's hierin hebben te maken met het veranderen van OSvectoren. In contrast met het eerste hoofdstuk gaat het hier om utility's. De in dit hoofdstuk verwerkte informatie is trouwens ook niet mis. Wat bijvoorbeeld te denken van het volgende:
"N.B. Het is door de PTT niet toegestaan je computer zonder tussenkomst van een modem met het telefoonnet te verbinden. Je mag dus niet:
- de audio-uitgang van je computer met het gele draadje verbinden
- de ingang met blauw verbinden
- massa met groen verbinden".

* Oneliners

Dit kleine hoofdstukje bevat een eenentwintigtal oneliners. Er zijn bijvoorbeeld: het spelletje YIP, een decimaal binair converter, een ESP tester en een programma dat een Fibonacci-reeks berekent. Daar had ik trouwens nog nooit van gehoord. Misschien kan een van onze 'wiskundigen' mij even uit de droom helpen voor wat betreft de toepassingen van deze reeks.

* Systeemadressen

De naam zegt het al. Dit is een lijst met adressen uit zowel zeroblock als C- en F-ROM.

* Afkortingen
* Hexadecimaal decimaal conversie tabel

Ook deze beiden spreken voor zich.

Al met al een waardige uitbreiding van de Atom-literatuurlijst volgens mij.
(te bestellen door overmaking van f.29,50 + f3,50 verzendkosten op giro 1241100 t.n.v. Elektuur B.V. onder vermelding van de titel v.h. boekje.)